Aplicação de XML para intercâmbio de documentos complexos

William Mansano

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Aplicação de XML para intercâmbio de documentos complexos

William Mansano[1]

Novembro - 2002

Banca Examinadora:

- Prof. Dr. Célio Cardoso Guimarães (Orientador)
- Prof. Dr. Ivan Luiz Marques Ricarte
  Faculdade de Engenharia Elétrica e de Computação - UNICAMP
- Prof.ª Dr.ª Claudia Maria Bauzer Medeiros
  Instituto de Computação - UNICAMP
- Prof. Dr. Luiz Eduardo Buzato
  Instituto de Computação - UNICAMP

[1]**Este trabalho foi realizado com suporte financeiro do CNPQ.**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Mansano, William

M317a Aplicação de XML para intercâmbio de documentos complexos / William Mansano -- Campinas, [S.P. :s.n.], 2002.

Orientador : Célio Cardoso Guimarães

Co-orientador: Nelson Castro Machado

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. XML (Linguagem de programação de computador). 2. Intercâmbio eletrônico de dados. 3. Sistemas de recuperação da informação. I. Guimarães, Célio Cardoso. II. Machado, Nelson Castro. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 18 de dezembro de 2002, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. Ivan Luiz Marques Ricarte
FEEC - UNICAMP

Profa. Dra. Claudia Maria Bauzer Medeiros
IC - UNICAMP

Prof. Dr. Célio Cardoso Guimarães
IC - UNICAMP

Aplicação de XML para intercâmbio de documentos complexos

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por William Mansano e aprovada pela Banca Examinadora.

Campinas, 19 Dezembro 2002.

Prof. Dr. Célio Cardoso Guimarães
(Orientador)

Prof. Dr. Nelson Castro Machado
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Agradecimentos

Quero agradecer às pessoas que me acolheram e forneceram subsídios para que eu alcançasse o meu objetivo.

Ao meu orientador Célio Cardoso Guimarães e co-orientador Nelson Castro Machado, pela orientação, dedicação e empenho.

À Banca Examinadora, Ivan Luiz Marques Ricarte, Claudia Maria Bauzer Medeiros e Luiz Eduardo Buzato.

Ao Instituto de Computação da UNICAMP, pelo bom ambiente de trabalho que oferece para a realização dessa dissertação.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela ajuda financeira recebida durante o curso.

À meus pais, irmão e familiares pela paciência e compreensão.

Aos amigos que me acompanharam neste processo.

E, principalmente, a DEUS.

"A vida só pode ser compreendida olhando-se para trás;
mas só pode ser vivida, olhando-se para frente."
(Soren Kierkergaard)

Resumo

O padrão XML (eXtensible Markup Language) está se tornando o principal formato para a troca de informações entre organizações e para a publicação de dados na Web, motivados pela necessidade de interoperabilidade entre as diferentes estruturas de armazenamento dessas informações. Este trabalho tem como objetivo fazer um overview de alguns padrões relacionados com a meta-linguagem XML e sua aplicação no projeto de uma aplicação para a criação de documentos XML e a sua conversão para documentos em diversos formatos (PS, PDF, RTF, HTML), voltada para a publicação de catálogos de universidades e a sua distribuição na Web.

Abstract

Standard XML (eXtensible Markup Language) is becoming the main format for information exchange between organizations and for the publication of data in the Web, motivated by the necessity of interchange of different structures of information storage. The objective of this work is to make an overview of standards related with the metalanguage XML and its application in the design of an application for creation of a complex XML document and its conversion to diverse formats (PS, PDF, RTF, HTML), for a university catalogue publication distribution in the Web.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A publicação e a troca de informação são necessidades vitais na maioria das organizações. Por exemplo, a troca de informação realizada entre empresas, filiais, fornecedores e clientes, e a divulgação de informação no comércio eletrônico via Internet. Na troca de informação existe o problema da interoperabilidade entre as diferentes arquiteturas utilizadas por cada organização, dificultando e tornando cada vez mais complexo esse processo.

XML (eXtensible Markup Language) [7], uma linguagem proposta pela W3C (World Wide Web Consortium), está se tornando o formato padrão para publicação e troca de informação, principalmente na Web (World Wide Web): XML provê estruturação de dados independente de plataforma, ou do modelo de dados em que as informações serão representadas. Existem vários padrões relacionados à linguagem XML e alguns deles serão apresentados nesse trabalho, como o DTD (Document Type Definition) que define as regras de estruturação no documento XML, e algumas linguagens de estilo para a apresentação das informações contidas no documento XML, como DSSSL (Document Style Semantics and Specification Language) e XSL (Extensible Style Language).

Será descrito um modelo para o intercâmbio de informações utilizando a tecnologia XML na estruturação dos dados, na criação de documentos XML e sua conversão para documentos de diferentes formatos, como HTML - HyperText Markup Language, Rtf - Rich Text Format,

Ps - PostScript e Pdf - Portable Document Format. O objetivo é a publicação de livros ou catálogos, e a sua disponibilização na Web através do formato HTML. Uma aplicação típica desse modelo, que foi tomada como elemento motivador desse trabalho, é a publicação dos catálogos de graduação e pós-graduação da Universidade Estadual de Campinas (Unicamp) e sua disponibilização na Web.

O exemplo utilizado para a aplicação desse modelo se baseia na elaboração de documentos acadêmicos, consistindo na geração e controle dos catálogos de graduação e pós-graduação dos cursos ministrados na Unicamp, suas habilitações e modalidades, o elenco das disciplinas que os integram com suas ementas e pré-requisitos, os prazos mínimo e máximo para integralização dos cursos, bem como a sugestão das Unidades para cumprimento dos currículos plenos.

A metodologia atualmente utilizada na geração desses catálogos apresenta várias desvantagens, pois não há uma automação na troca de informações entre os coordenadores dos cursos e a administração da Universidade: esta troca é realizada manualmente através de documentos enviados em papel para a administração, que digita novamente todas as informações fornecidas. Tendo as informações em um formato digital, ainda são realizados vários processamentos para verificar a formatação e correção do conteúdo dos catálogos. Outras desvantagens são o formato interno "ad-hoc" utilizado para representar a informação de forma semi-estruturada e a redundância de informação presente em várias partes ao longo dos catálogos de cursos.

Com o objetivo de resolver o problema de interoperabilidade na troca de informação entre os coordenadores dos cursos e a administração da Universidade será definido um padrão de documentos XML para a estruturação das informações e a automação da geração e publicação dos catálogos de graduação e pós-graduação, seguindo o modelo de intercâmbio de informações, incluindo: (i) a geração dos documentos dentro do padrão XML definido para a aplicação através de formulários Web a serem utilizados pelas unidades responsáveis da Unicamp, (ii) a conversão dos documentos XML para os formatos de impressão (RTF, PS e PDF) e Web (HTML), a partir de estilos definidos em linguagens de estilo específicas para o

padrão XML (DSSSL ou XSL).

O diagrama da figura 1.1 ilustra cada etapa do modelo desde o processo de criação de documentos XML até sua conversão para a publicação em diferentes formatos.

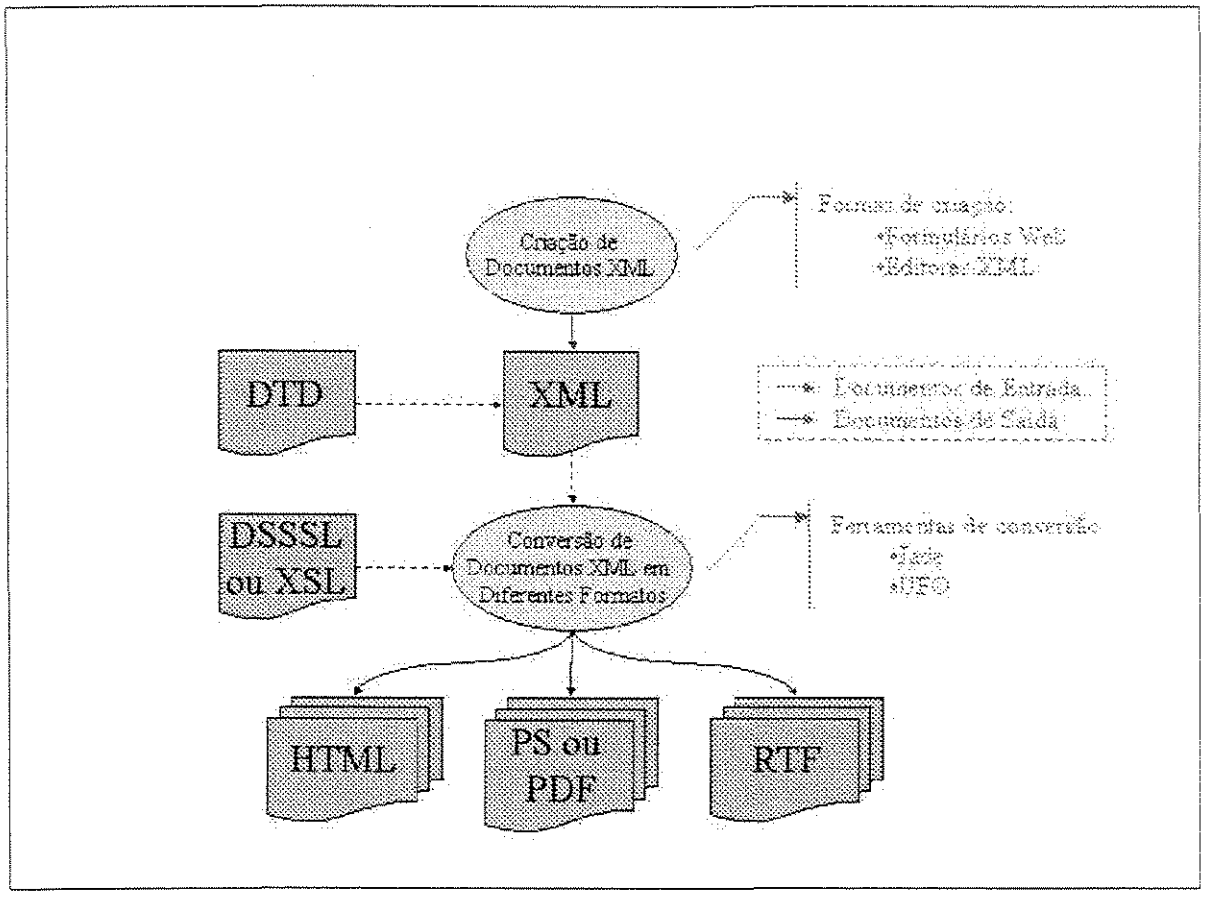

Figura 1.1: Diagrama do Modelo de Intercâmbio.

Inicialmente será necessária a criação de documentos dentro de um padrão XML pré-estabelecido para a aplicação prática do modelo. Para se definir um documento padrão XML é necessário criar um DTD (Document Type Definition) [19] que represente as definições desejadas para a aplicação do modelo. O DTD define a estruturação dos elementos nos documentos XML: quais elementos e atributos poderão ser utilizados e qual a organização hierárquica entre eles.

Na definição do DTD foram utilizadas algumas especificações do padrão DocBook [35], que

provê um sistema para escrita de documentos estruturados usando SGML ou XML. DocBook provê DTDs contendo especificações para o conteúdo de um livro, como por exemplo para o título, seções, parágrafos e referências bibliográficas do livro, podendo ser publicado em formatos para impressão e para a Web, através de estilos descritos especificamente para o padrão DocBook. Devido às necessidades encontradas na nossa aplicação foi preciso criar vários outros elementos, e consequentemente outros estilos de publicação que representassem características particulares da aplicação.

A criação de documentos no padrão XML poderá ser realizada de duas maneiras: através de editores de documentos XML ou de formulários desenvolvidos especificamente para a aplicação. Editores XML oferecem recursos para criação e análise do documento, mas os usuários terão que conhecer a estruturação do documento em XML para que utilizem os elementos corretos em suas devidas posições. Na utilização de formulários os usuários não precisam ter conhecimento de XML, mas há a necessidade de desenvolver formulários, que ao serem processados gerem a estrutura necessária do documento XML.

A próxima etapa consiste na geração dos documentos finais a partir do conjunto de documentos XML, usando a ferramenta "Jade" [11]. Para gerar os documentos finais desejados (HTML, RTF, PS ou PDF) é necessária a definição de estilo dos documentos seja no formato DSSSL (Document Style Semantics and Specification Language) [22], ou XSL (Extensible Stylesheet Language) [2]. Eles especificam a visualização do documento final, tanto para a Web (HTML) quanto para documentos do tipo texto (RTF, PS e PDF) para impressão.

O formato HTML [28], em princípio é independente de plataforma, já que esses documentos serão interpretados por um navegador Web. Os documentos HTML podem ser gerados pela ferramenta Jade, disponível para diversas plataformas, utilizando uma especificação própria do estilo DSSSL.

Para a geração de documentos nos formatos PS e PDF, a ferramenta Jade faz uso de recursos externos como o "Latex", para a criação de documentos no formato DVI (Device Independent), e posteriormente, a partir do DVI a criação de documentos PS e PDF.

Esse trabalho está organizado da seguinte forma: a seção 2 introduz conceitos básicos

sobre o padrão XML e as demais estruturas necessárias para sua utilização. A seção 3 descreve abordagens relacionadas a este trabalho. A seção 4 descreve duas formas de criação de documentos XML: a utilização de editores XML, dentre vários disponíveis em diversas plataformas e que serão revistos, e a utilização de formulários Web e descreve uma proposta de conversão das informações estruturadas em XML para documentos de diferentes formatos. A seção 5 mostra os resultados obtidos com a aplicação do modelo descrito nesse trabalho sobre os catálogos de cursos utilizados na Unicamp. A seção 6 apresenta considerações finais.

Capítulo 2

Conceitos sobre XML e Padrões Correlatos

Neste capítulo serão apresentados os principais conceitos do padrão XML utilizados para a criação de um modelo de documentos, assim como padrões correlatos utilizados na transformação desses documentos, visando a sua aplicação na geração e publicação dos catálogos da Unicamp.

2.1 A Linguagem XML - Extensible Markup Language

XML [7], ou Extensible Markup Language, é uma meta-linguagem derivada da SGML (Standard Generalized Markup Language) [18] que provê mecanismos para a definição de tags[1] que identificam a estrutura de um documento, possibilitando a criação de diferentes modelos de documentos, ou seja, a XML e SGML são linguagens genéricas para definir a formatação de documentos e por isto são também chamadas de "Meta-linguagens de marcação". Hoje acredita-se que XML será responsável por uma revolução na Web, graças às características para criar sistemas de documentos com muito mais funcionalidade do que hoje é

[1]São marcações no texto que aparecem aos pares e são cercados pelos sinais de menor ("<") e maior (">"). Tags indicam o início e o fim de uma característica da informação ou um comando em um documento XML.

permitido através do uso da HTML (HyperText Markup Language) [28].

A linguagem predecessora, SGML, foi criada por pesquisadores da IBM com o objetivo de construir um sistema portável para o intercâmbio e manipulação de documentos, sendo independente de sistema operacional ou formatos de arquivos. O sistema então chamado de "Marcação Generalizada" (Generalized Markup) possui como principais objetivos:

- Descrever a estrutura do documento e demais atributos através de marcações em seu conteúdo, sem o processamento do documento.
- Definir rigorosamente sua marcação para que vários sistemas possam processar documentos SGML.

Em 1996 foi criada a XML como uma versão em menor escala da SGML, com a idéia de aproveitar o poder da SGML para aplicações voltadas à Web e por outro lado diminuir a complexidade da validação sintática de documentos SGML. O objetivo principal, além de facilitar a definição de tipos de documentos é a transmissão e compartilhamento dos arquivos pela Web. Em 1998, XML tornou-se uma especificação formal da W3C (World Wide Web Consortium), que é a organização responsável por especificar padrões para a Web.

Como SGML, XML é uma meta-linguagem para a descrição de padrões de marcação, ou seja, XML não possui um conjunto pré-definido de tags ou elementos como ocorre em HTML, podendo definir o significado dos dados através de tags de acordo com o domínio dos dados e da aplicação utilizada.

Objetivos do desenvolvimento da XML

O principal objetivo buscado pela XML é possibilitar que documentos sejam distribuídos e processados na Web da mesma forma que hoje é possível com HTML. Além disso, teve como metas a facilidade de implementação e a interoperabilidade com SGML e HTML, possibilitando a criação de uma nova geração de aplicações para Web, seja para visualização ou para manipulação de dados.

A seguir apresentamos um resumo extraído da especificação da XML:

1.	Ser francamente utilizável pela Internet.
2.	Ser capaz de suportar uma grande variedade de aplicações.
3.	Ser compatível com SGML.
4.	Tornar mais fácil escrever programas que processem documentos XML.
5.	Os documentos XML devem ser legíveis por seres humanos.
6.	O projeto da XML deve ser formal e conciso.
7.	Os documentos XML devem ser facilmente criados.
8.	A concisão na marcação em XML é de importância menor.

Tabela 2.1: Objetivos do desenvolvimento da XML.

SGML também provê estruturação da informação, mas sua implementação é muito complexa para um browser Web. Sendo derivada da SGML, XML permite que sistemas que suportam SGML tenham a capacidade de ler documentos XML. Mas para usar e processar XML não é necessário um sistema capaz de entender toda a generalidade de SGML.

Em HTML a semântica e o conjunto de tags são fixos, não sendo possível ampliar as definições HTML com novos tags e trazendo variações de apresentação na Web. XML provê uma facilidade de definir tags e a estrutura relacional entre eles. Não possui um conjunto de tags pré-definidas, não há uma semântica fixa: toda a semântica de um documento XML será definida pela aplicação que irá processá-los.

XML fornece um padrão para codificação do conteúdo e semântica para uma grande variedade de casos, variando de aplicações simples (como um registro estruturado ou um breve documento) a complexas (objetos com atributos e métodos, conteúdo de site na Web, descrição de interfaces gráficas com o usuário, links entre informações e pessoas na Web, etc).

2.1.1 Necessidade da Utilização de XML

HTML também é derivada da linguagem SGML, mas é uma instância criada apenas para a composição e apresentação de documentos estruturados na Web, podendo combinar texto, imagens e botões em um navegador para a Internet, sem se preocupar com o intercâmbio de informações pela Web. Esta inovação foi a grande responsável pela popularização da Internet, permitindo a usuários de computadores de todo o mundo o acesso a inúmeros documentos e informações que podiam ser lidas em seus computadores onde quer que estejam.

Atualmente pessoas e companhias querem websites que possam receber pedidos de seus clientes, transmitir históricos médicos, pôr em funcionamento a partir de qualquer lugar fábricas e equipamentos científicos. HTML não foi concebida para tarefas como estas. Sua simplicidade foi essencial para que as pessoas pudessem dominá-la e daí a rápida expansão e popularização da Web, mas já não atende às necessidades atuais do mercado.

Um dos principais problemas da linguagem HTML é a mistura da estruturação de um documento com a forma de apresentação do mesmo. Uma alternativa para esse problema é a utilização de *"folhas de estilo"* CSS (Cascading Style Sheets) [5] que permitem especificar separadamente da marcação HTML a forma de apresentação de cada elemento de um documento. Sucessora da HTML, a linguagem XHTML (Extensible HyperText Markup Language) [27] vem como uma solução para o principal problema da HTML: (i) segue o padrão da meta-linguagem XML, com uma rígida estruturação e aninhamento dos seus tags e, (ii) separa a forma de apresentação do documento da estruturação de seu conteúdo através do uso de CSS.

Ao contrário de SGML, XML não requer um DTD para cada documento, embora isto seja altamente desejável, pois um DTD define uma gramática para os elementos e atributos do documento. Em SGML a análise sintática para a validação de documentos pode ser muito complexa sendo uma das principais razões para XML adotar uma forma mais simples, um aninhamento completo dos elementos, e requisitos rígidos para valores de atributos dos elementos de um documento XML.

XML não guarda apenas informações, mas também interpreta o que elas significam, através do uso de tags para definir o que é a informação e não como ela deve ser mostrada. XML aperfeiçoará uma ampla faixa de aplicativos da Internet, desde as redes de fornecimento da indústria de automação até uma nova classe de programas da Internet, aos quais os usuários poderão ter acesso no escritório, em casa ou na estrada.

HTML, XHTML e XML são padrões internacionais originados no W3C (World Wide Web Consortium) e sancionados pelas organizações ANSI (*American National Standards Institute*) e ISO (*International Organization for Standardization*), sendo portanto independentes de fabricantes de hardware e software.

Serão citadas a seguir algumas das principais vantagens da XML para os vários tipos de aplicações existentes [24].

Principais vantagens da XML:

- *Estruturação:* XML fornece uma representação estruturada de dados dando uma semântica à informação, que pode ser implementada a partir de um grande número de aplicações. É apropriada para distribuição pela Web, garantindo que os dados estruturados serão uniformes e independentes de aplicações ou distribuidores.

- *Flexibilidade:* XML separa a estrutura do conteúdo de um documento de sua apresentação final, permitindo dar maior importância para o conteúdo e estruturação da informação, sem se preocupar com a sua apresentação. O mesmo documento pode ser apresentado de várias formas, sejam elas em um monitor de um computador pessoal ou numa impressora.

- *Intercâmbio de dados:* A XML permite a combinação de dados estruturados oriundos de diferentes fontes. Componentes podem ser utilizados para obter informações na Web, integrando os dados oriundos de diferentes servidores importando-os diretamente para seus aplicativos. Esses dados podem ser distribuídos a clientes ou a outros servidores para posterior agregação, processamento e distribuição da informação.

- *Buscas:* Com XML os dados podem ser facilmente categorizados a partir da adoção de um padrão comum (por exemplo um livro pode conter: autor, título, assunto, etc), que irá facilitar a busca de informações nos documentos XML, permitindo que a aplicação final realize uma consulta mais consistente.

- *Automação:* A partir de informações mantidas em websites, ou em bases de dados já existentes, podem ser automaticamente produzidos arquivos XML de fácil leitura por qualquer aplicação.

- *Simplicidade:* XML provê tanto para programadores como autores de documentos um ambiente amigável do ponto de vista da computação. O rígido conjunto de regras da XML ajuda a tornar documentos mais fáceis de serem entendidos. A sintaxe de documentos XML contém um conjunto relativamente pequeno de regras, possibilitando aos desenvolvedores uma rápida iniciação na linguagem. DTDs podem ser desenvolvidos através de processos padronizados, baseados nas estruturas dos documentos desejados.

- *Extensibilidade:* A linguagem XML é extensível de dois modos: primeiro, ela permite aos desenvolvedores a criação de seus próprios DTDs, criando efetivamente conjuntos de tags extensíveis que podem ser usadas para múltiplas aplicações, ou seja, no momento em que um DTD é adotado em uma organização, surge uma nova habilidade na busca e processamento dos dados de maneira independente do *front-end*, podendo ser apresentados sob diversas formas como browsers ou outras aplicações.

  Segundo, XML está sendo estendida com uma série de padrões adicionais que acrescentam propriedades de estilo, ligações e referenciação ao conjunto básico de capacidades da linguagem, dando um conjunto maior de opções além da linguagem XML em si.

- *Interoperabilidade:* XML pode ser utilizada sobre uma grande variedade de plataformas e interpretada por uma grande variedade de ferramentas. Devido ao comportamento consistente das estruturas de documentos, os navegadores que os interpretam podem ser desenvolvidos a um custo relativamente baixo.

- *Abertura:* O processo utilizado na criação da XML é completamente aberto e disponível na Web. A W3C disponibiliza prontamente os avanços obtidos, possibilitando o acompanhamento de seus progressos.

  Companhias podem criar documentos XML que se comportem de maneira específica no que tange às suas aplicações, tendo os dados disponíveis para quaisquer outras aplicações. Desenvolvedores de aplicações podem dividir tarefas entre várias ferramentas, possivelmente até de diferentes aplicações concorrentes, permitindo assim que eles operem no mesmo conjunto de estruturação de dados. A linguagem não impede a criação de formatos exclusivos, mas a sua abertura é uma de suas principais vantagens

O formato XML é interessante não só para a Internet como também para ambientes Intranet em grandes corporações, pois permite interoperabilidade a partir do uso de um formato flexível, aberto e padronizado, com novas formas de acessar arquivos legados e distribuir os dados a clientes Web. Assim, as aplicações podem ser construídas mais rapidamente, com maior facilidade de manutenção, e permitindo múltiplas formas de visualização dos dados estruturados.

Atualmente vários pacotes de software livres e comerciais estão sendo disponibilizados para manipulação de documentos XML.

2.1.2 Estrutura de Documentos XML

Um documento XML é composto de declarações, elementos, comentários, referências a caracteres e instruções de processamento; todos são indicados no documento por uma marcação explícita [34]. Essas estruturas devem estar aninhadas apropriadamente como uma hierarquia em forma de árvore, sempre iniciando por um elemento *raiz* no documento XML.

Documentos XML são compostos de marcações e conteúdos. Existem seis tipos de marcações: elementos, referências a entidades, comentários, instruções de processamento, seções marcadas e DTDs (Document Type Definition) que definem regras para a utilização dessas marcações, que serão detalhadas a seguir.

Elementos

Elementos são a forma mais comum de marcação em um documento XML, que é composto por pelo menos um elemento chamado de elemento *raiz*. Delimitados pelos sinais de menor ("<") e maior (">"), a maioria dos elementos identifica a natureza do conteúdo que eles envolvem. Alguns elementos podem ser vazios: neste caso eles não possuem nenhum conteúdo. Se um elemento não é vazio, sua extensão é delimitada por um tag inicial e um tag final contendo o nome do elemento; o conteúdo é inserido entre esses tags (ex.: *<autor>João</autor>*).

Elementos não vazios podem conter elementos filhos, que devem ser declarados em uma lista de sub-elementos em sua definição, e também podem conter caracteres em seus conteúdos, podendo ocorrer as duas situações simultaneamente, ou seja, um elemento pode conter sub-elementos e caracteres em seu conteúdo, ou ocorrer separadamente com apenas sub-elementos ou apenas caracteres entre os tags do elemento. Esta definição é recursiva.

Elementos vazios possuem uma sintaxe modificada: enquanto a maioria dos elementos em um documento envolve algum conteúdo, os elementos vazios são simplesmente marcadores onde alguma coisa ocorre (por exemplo, uma separador horizontal para o tag <hr> em HTML). O final *"/>"* na sintaxe modificada indica a um programa que processará o documento XML que o elemento é vazio e um tag final correspondente não deve ser procurada (ex.: <br/>).

Atributos

Atributos são pares de valores com nome que ocorrem dentro de tags iniciais e após o nome do elemento. Por exemplo:

* *

Esse é um elemento chamado *div* cujo atributo *classe* possui o valor *prefácio*. Em XML, todos os valores de atributos devem estar entre aspas.

Valores possíveis para atributos e suas formas de definição serão detalhados na seção referente ao DTD.

Referências a Entidades

Alguns caracteres foram reservados para identificar elementos. O sinal de menor ("<") e maior (">"), por exemplo, identificam o início e fim de um tag no documento XML. Para inserir esses caracteres como conteúdo, deve haver uma alternativa para representá-los. Em XML, entidades são usadas para representar esses caracteres especiais. As entidades também são usadas para referenciar um texto frequentemente repetido ou alterado e incluí-lo no conteúdo do documento através de arquivos externos.

Cada entidade deve ter um nome único. Para usar uma entidade, você simplesmente a referencia pelo nome,precedido do "E" comercial (&) e terminando com um ponto-e-vírgula (;).

Por exemplo, a entidade *lt* insere um literal "<" em um documento. A cadeia de caracteres *<element>* pode ser representada em um documento XML como *<element>*, por exemplo.

Uma forma especial de referência a entidades, chamada de referência a caracteres, pode ser usada para inserir arbitrariamente caracteres *Unicode* em um documento. Este é um mecanismo para inserir caracteres que não podem ser diretamente digitados pelo teclado.

Referências a caracteres podem ter uma das duas formas: referências decimais &#8478, e referências hexadecimais &#x211E. Neste exemplo, ambas se referem ao caracter *Unicode* número U+211E.

Comentários

Comentários iniciam com a seqüência de caracteres "<!- -" e terminam com "- ->"; podem conter qualquer dado, exceto o literal "- ->", usado para identificar o término do comentário. Comentários podem ser colocados entre tags ou em qualquer outro lugar dentro do documento XML. Eles não fazem parte do conteúdo textual do documento XML. Um processador XML não precisa reconhecê-los para a aplicação que será utilizada.

Instruções de Processamento

Instruções de processamento são formas de fornecer informações a uma aplicação. Assim como os comentários, elas não são textualmente parte de um documento XML, mas neste caso o processador XML deverá reconhecê-las para a aplicação em questão.

As instruções de processamento possuem a seguinte forma: *<?nome dados?>*. O nome, chamado de alvo da instrução de processamento, identifica a instrução de processamento na aplicação. As aplicações processam somente os alvos que elas reconhecem e ignoram as outras instruções de processamento. Qualquer dado que segue o alvo da instrução de processamento é opcional para a aplicação que reconhece o alvo. Os nomes usados nas instruções de processamento podem ser declarados como notações em seu respectivo DTD, a fim de identificá-los formalmente.

Os nomes das instruções de processamento que iniciam com o alvo *xml* são reservados para a padronização da XML.

Exemplo: *<?xml version="1.0"?>*.

Seções CDATA

Em um documento XML, uma seção CDATA instrui o analisador XML para ignorar a maioria dos caracteres de marcação existentes nessa seção.

Considere um código fonte em um documento XML. Ele pode conter caracteres que o analisador XML iria normalmente reconhecer como marcação ("<" e "&", por exemplo). Para que estes caracteres não sejam reconhecidos como caracteres especiais, uma seção CDATA pode ser usada, como segue o exemplo.

Exemplo:

```
<![CDATA[
        *p = &q;
        b = (i <= 3);
]]>
```

Entre o início da seção *"<![CDATA["* e o fim da seção *"]]>"*, todos os dados de caracteres

são passados diretamente para a aplicação, sem interpretação do analisador XML. Elementos, referências a entidades, comentários e instruções de processamento são todos irreconhecíveis e os caracteres que os compõem são passados literalmente para a aplicação.

A única cadeia de caracteres que não pode ocorrer durante uma seção CDATA é *"]]>"*, usado para identificar o término de uma seção.

Exemplo de documento XML

```
<documento>
   <livro ano="2002">
      <autor>
         <nome>José</nome>
         <sobrenome>Saramago</sobrenome>
      </autor>
      <titulo>A Caverna</titulo>
      <editora>
         <nome>Companhia das Letras</nome>
         <endereco>R. Bandeira Paulista, 702 - São Paulo - SP</endereco>
      </editora>
   </livro>
</documento>
```

Figura 2.1: Exemplo de documento XML de uma biblioteca.

O documento XML da figura 2.1 representa um livro de uma biblioteca. O documento é iniciado com o elemento raiz *documento*. O elemento *livro* identifica o tipo de documento que está sendo descrito e a data de sua publicação pelo atributo *ano*, contendo o autor do livro no elemento *autor*, o título do livro no elemento *titulo*, e a editora no elemento *editora*. O autor do livro é dividido em nos elementos *nome* e *sobrenome*. E a editora do livro inclui os elementos *nome* e *endereco*. Os elementos finais *titulo*, *nome*, *endereco*, *nome* e *sobrenome* contém os dados reais do livro.

Dada a discussão precedente de declarações de tipos, conclui-se que alguns documentos

XML são válidos e outros não. Existem duas categorias de documentos XML, que serão detalhados a seguir: bem formados e válidos.

Documentos XML Bem Formados

Um documento é chamado de *bem formado* (*"well formed"*) se ele obedece à sintaxe da XML. Um documento que inclui seqüências de caracteres e de elementos (tags) que não podem ser analisados ou são inválidos não podem fazer parte de um documento XML bem formado.

Além disso, um documento XML bem formado deve atender às seguintes condições:

1. Deve possuir pelo menos um elemento.

2. O elemento raiz não pode estar contido dentro de nenhum outro elemento.

3. A instância do documento deve seguir a especificação gramatical de documentos XML. Em particular, algumas construções de tags são permitidas somente em locais específicos. O documento não é bem formado se tais construções ocorrerem em outros locais, mesmo que o documento esteja bem formado nos demais casos.

4. O texto de substituição para todas as entidades referenciadas dentro de uma declaração de elemento deve consistir de zero ou várias declarações de elementos completos (contendo tags iniciais e finais), ou seja, nenhuma entidade usada no documento pode consistir somente de uma parte de uma declaração de elemento (apenas um tag inicial ou final), mas sim da declaração completa de um ou mais elementos.

5. Nenhum atributo pode aparecer mais do que uma vez no mesmo tag inicial.

6. Valores de atributos devem ser cadeias de caracteres, e não podem conter referências a entidades externas.

7. Elementos não-vazios devem ser apropriadamente aninhados, ou seja, tags iniciais e finais de um sub-elemento devem estar contido dentro dos limites de um único elemento

"pai", formando uma hierarquia de árvore com o elemento "pai" (raiz da sub-árvore) contendo sub-elementos completos (folhas da sub-árvore).

8. Todas as entidades devem ser declaradas, exceto as seguintes: amp (&), lt (<), gt (>), apos (') e quot ("), que já estão pré-definidas pela linguagem XML.

9. Uma entidade binária (contém informações não textuais, como por exemplo, o conteúdo de uma figura que é de caráter binário) não pode ser referenciada explicitamente durante o fluxo do conteúdo, essa entidade pode ser usada somente através de um atributo declarado como ENTITY ou ENTITIES.

10. No texto ou em entidades parâmetro não são permitidos recursividade, direta ou indiretamente.

Mesmo sem um DTD o processador XML deve verificar se um documento XML é bem formado. Se caso contrário, não poderá ser um documento XML. Isto significa que não há documento XML que não seja bem formado e os processadores XML não devem fazer nada com tais documentos.

Essa característica é boa para aplicações Web, por que elas não necessitam conhecer a estrutura do DTD usado para gerar o documento XML.

Documentos XML Válidos

Um documento XML *"válido"* é um documento XML bem formado contendo um DTD (ou uma referência para um DTD) obedecendo às restrições das definições presentes nesse DTD (como por exemplo: se a seqüência e aninhamento dos elementos é válida, se atributos requisitados são fornecidos, se valores dos atributos são do tipo correto, etc).

2.2 DTD - Document Type Definition

Um DTD [19] para um documento XML provê a especificação de uma lista de elementos, atributos, comentários, notas e entidades que podem estar contidos no documento, indicando também o relacionamento entre eles, como tags podem ser usados, em que ordem devem aparecer, quais tags podem aparecem dentro de outros, quais possuem atributos, etc. Em outras palavras, um DTD é um conjunto de regras de sintaxe para o uso de tags em documentos XML, podendo ser considerado como a gramática de uma linguagem markup. A declaração de um elemento e de seus atributos e as demais declarações descritas em um DTD, como também na especificação XML seguem a notação gramatical EBNF (Extended Backus-Naur Form) [9] [17].

Também existem outras possibilidades para a definição das regras de documentos XML, como através de XML Schema [15] e a utilização de RDF (Resource Description Framework) [8]. Mas pela simplicidade de manipulação, este trabalho utilizou DTD na definição das regras para os elementos e atributos de um documento XML.

Devido a XML não ser realmente uma linguagem de programação, mas na verdade um sistema padrão (meta-linguagem) para a definição de diferentes estruturas, ela não possui um DTD universal a exemplo de HTML. Cada organização que desejar utilizar XML para a troca de informação poderá definir seu próprio DTD.

Para uma dada aplicação é provável não ter significado tags que ocorrem em uma ordem completamente arbitrária. Para que um documento XML tenha realmente um significado, e possa ser apresentado através de um padrão de estilo ou que seja processado por uma aplicação, devem haver restrições na sequência e aninhamento de seus tags. Nos DTD's estas restrições expressam a regulamentação do uso de tags nos documentos de uma dada aplicação.

Mais genericamente, as declarações em um DTD representam meta-informações para serem analisadas a respeito do seu conteúdo. Meta-informações incluem as sequências e aninhamentos permitidos para tags, valores de atributos, seus tipos e padrões, nomes de arquivos externos referenciados e se podem ou não conter dados XML, o formato de algum

dado externo "não-XML" e as entidades que podem ser encontradas.

Antes de detalharmos a sintaxe relativamente complexa de um documento dtd, vamos apresentar um exemplo simples:

2.2.1 Exemplo de DTD:

```
<!ELEMENT documento (livro | artigo)>

    <!ELEMENT livro (autor+, titulo, editora)>
        <!ATTLIST livro ano CDATA>

    <!ELEMENT artigo (autor+, titulo, ano?)>
        <!ATTLIST artigo tipo CDATA>

        <!ELEMENT autor (nome?, sobrenome)>

            <!ELEMENT nome (#PCDATA)>

            <!ELEMENT sobrenome (#PCDATA)>

        <!ELEMENT titulo (#PCDATA)>

        <!ELEMENT editora (nome, endereco)>

            <!ELEMENT endereco (#PCDATA)>

        <!ELEMENT ano (#PCDATA)>
```

Figura 2.2: Exemplo de DTD de uma biblioteca.

O DTD da figura 2.2 é referente ao documento XML mostrado na figura 2.1 representando um livro de uma biblioteca. Nesse DTD o elemento *documento* pode conter um livro ou um artigo. O elemento *livro* contém um ou mais elementos *autor*, um *titulo*, e um elemento *editora* nesta ordem e possui um atributo *ano*. O elemento *artigo* é similar ao elemento *livro*,

mas contém o elemento *ano* que é opcional, ao invés do elemento *editora*. Um *artigo* possui um atributo *tipo*. Um elemento *editora* contém os elementos *nome* e *endereco*, e um *autor* possui um elemento opcional *nome* e deve conter um elemento *sobrenome*. Foi assumindo que *titulo*, *nome*, *endereco*, *nome* e *sobrenome* são todos do tipo *PCDATA*, isto é, valores de string.

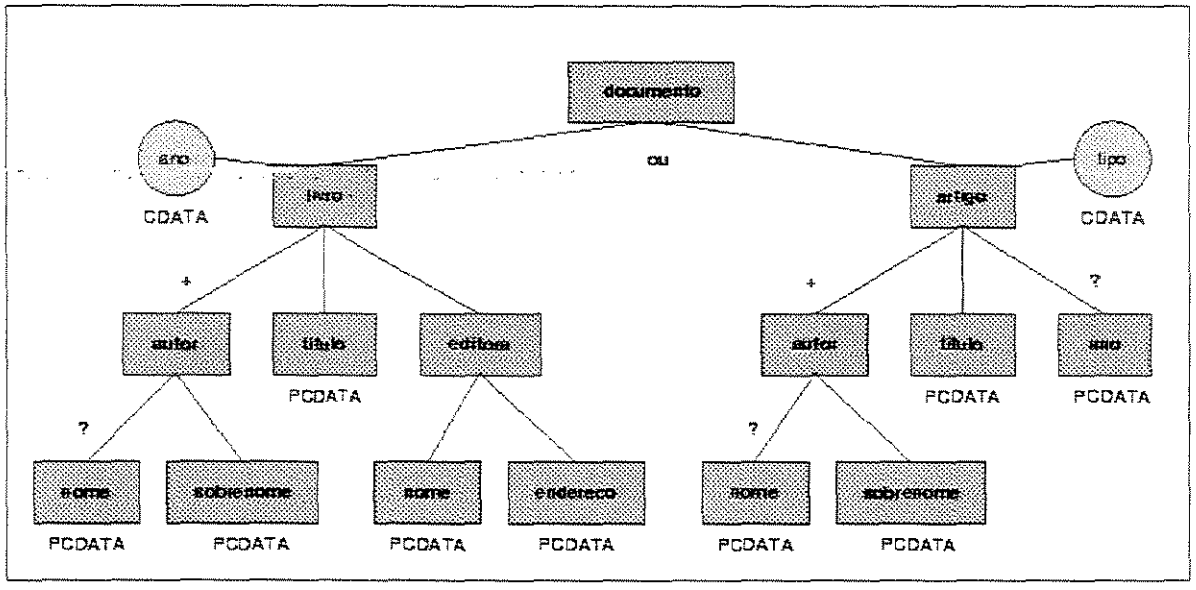

Figura 2.3: Diagrama de hierarquia do DTD da figura 2.2.

O Diagrama da figura 2.3 representa a estrutura hierárquica dos elementos declarados no DTD da figura 2.2

Um DTD é definido apenas uma vez e pode ser definido para uso local. Trocas subseqüentes de dados podem referenciar uma cópia do DTD original armazenado localmente. Alternativamente, DTDs podem estar em localizações remotas (tal como um repositório centralizado de DTDs).

Se presente, o DTD deve ser a primeira especificação em um documento depois de comentários e instruções de processamento opcionais. O DTD identifica o elemento raiz do documento e pode conter declarações adicionais. Todos os documentos XML devem ter um elemento raiz único que contenha todo o conteúdo do documento. Declarações adicionais

podem vir de um DTD externo ou serem incluídas diretamente no documento XML que pode conter ambas (figura 2.4). Caso o DTD esteja separado, o documento XML deve conter instruções para poder encontrar o respectivo DTD, para ser utilizado por um analisador sintático de documentos XML, a fim de analisar gramaticalmente a correção de um documento XML.

Um elemento *<!DOCTYPE>* no começo do arquivo XML irá informar ao analisador sobre a localização do DTD e indicará qual será o elemento raiz do documento XML. Como mostra o exemplo da figura 2.4, o elemento raiz do documento será *biblioteca*:

```
<?xml version="1.0"?>
<!DOCTYPE biblioteca SYSTEM "biblio.dtd"[
   <!ELEMENT ulink (#PCDATA)>
      <!ATTLIST ulink URL CDATA #REQUIRED> ]>

<biblioteca>
   ...
   <ulink URL="www.biblioteca.com.br">Biblioteca Virtual</ulink>
   ...
</biblioteca>
```

Figura 2.4: Documento XML contendo a declaração de seu DTD.

Apenas a estrutura lógica de um documento XML é descrita pelo DTD, não sendo fornecida informação alguma sobre a forma de apresentação do documento. Deste modo, para uma biblioteca, por exemplo, é dada a estrutura de seus livros, artigos, editoras, etc, mas nada é definido a respeito do formato de apresentação destas informações, por exemplo, qual o tamanho e fonte de caracteres a serem utilizados em sua apresentação.

2.2.2 Declarações de Elementos

Declarações de elementos identificam os nomes dos elementos e a natureza do seu conteúdo. Um elemento pode ser definido em um DTD como um grupo de sub-elementos, PCDATA

(Parsed Character DATA), EMPTY ou ANY.

- *Grupo de sub-elementos*:

  São os elementos que podem conter sub-elementos em sua definição. Por exemplo:

  <!ELEMENT A (B, C)>

  Os sub-elementos podem ser agrupados em sequência ("and") ou podem ser uma escolha ("or") desses sub-elementos. A seguir serão dados exemplos desses dois tipos de agrupamentos:

  - *Sequência*:
    - O elemento A possui o elemento B seguido pelo elemento C:
      <!ELEMENT A (B, C)>
    - O elemento A possui uma sequência, incluindo um sub-grupo de escolha:
      <!ELEMENT A (B, (C | D), E>
  - *Escolha*:
    - O elemento A pode possuir o elemento B ou o elemento C:
      <!ELEMENT A (B | C)>
    - O elemento A possui uma escolha, incluindo um sub-grupo de sequência:
      <!ELEMENT A (B | C | (D, E))>

  Os seguintes operadores podem ser utilizados em um DTD para serem aplicados na definição de seus sub-elementos:

  - *Opcional (?)*:

    O sub-elemento B é opcional dentro do elemento A:

    <!ELEMENT A (B?, C)>
  - *Uma ou mais vezes (+)*:

    O sub-elemento C pode ocorrer uma ou mais vezes dentro do elemento A:

    <!ELEMENT A (B, C+, D)>

- *Zero ou mais vezes (*)*

  O grupo (B, C) pode ocorrer zero ou mais vezes dentro do elemento A, dessa forma o elemento A também pode ser um elemento vazio:

  <!ELEMENT A (B, C)*>

- *PCDATA*:

  Os elementos declarados do tipo PCDATA são aqueles elementos que podem conter apenas caracteres. Um exemplo de um elemento A que possui apenas texto em seu conteúdo, e que não pode conter sub-elementos:

  <!ELEMENT A (#PCDATA)>

  Uma variação bastante útil desses tipos de elementos PCDATA são os elementos *mistos* que podem conter tanto caracteres quanto sub-elementos e em qualquer ordem desejada. Por exemplo, o elemento A pode conter os sub-elementos B e C além de também conter caracteres em seu interior.

  <!ELEMENT A (#PCDATA | B | C)*>

- *EMPTY*:

  São os elementos que não possuem sub-elementos ou caracteres, ou seja, indica que o elemento não possui conteúdo (consequentemente não possui tag final). Mas esses elementos podem possuir atributos em seus tags.

  Exemplo de uma declaração de um elemento vazio A:

  <!ELEMENT A EMPTY>

- *ANY*:

  São os elementos que podem conter zero ou mais sub-elementos de qualquer tipo declarado no DTD, como também pode conter caracteres em seu interior, indicando que qualquer conteúdo é permitido para esse elemento.

  Exemplo de uma declaração de um elemento do tipo ANY:

<!ELEMENT A ANY>

O modelo de conteúdo *ANY* é algumas vezes útil durante a conversão de documentos, mas deve ser evitado ao máximo em um ambiente de produção, pois desabilita toda a verificação do conteúdo desse elemento.

Elementos definidos como um grupo de sub-elementos consiste em um elemento *não terminal* na linguagem. Os elementos definidos como PCDATA, EMPTY, ANY constituem os elementos *terminais* da linguagem.

- *Elemento não terminal*: <!ELEMENT A (B)>
- *Elemento terminal*: <!ELEMENT A (#PCDATA)>

Um exemplo de uma declaração típica de um elemento se parece com o seguinte exemplo:

<!ELEMENT livro (autor+, titulo, editora?)>

Esta declaração identifica o elemento nomeado como *livro*. O modelo de conteúdo define o que um elemento pode conter. Neste caso, um *livro* deve conter os elementos *autor*, *titulo* e pode conter *editora*. As vírgulas entre os nomes dos elementos indicam que esses elementos devem ocorrer na sequência apresentada. O elemento *autor* pode ser repetido mais de uma vez, mas deve ocorrer pelo menos uma vez no documento XML. O elemento *editora* é opcional, ou seja, pode estar ausente ou ocorrer somente uma vez. Um nome sem pontuação, como *titulo*, deve ocorrer somente uma vez.

As declarações em um DTD para os elementos usados em um modelo de conteúdo XML pode ser utilizado por um processador XML para verificar a validade de seus documentos.

Por exemplo, a definição de um elemento *paragrafo* dentro de um livro ou artigo pode ser representado pela seguinte expressão:

<!ELEMENT paragrafo (#PCDATA | lista)*>

Nessa definição o elemento *paragrafo* pode conter zero ou mais caracteres e elementos *lista* misturadas em qualquer ordem. Todos os modelos de conteúdo misto devem ter esta forma: #PCDATA deve vir primeiro e todos os elementos devem ser separados por barras verticais e o grupo inteiro deve ser opcional.

2.2.3 Declarações de Listas de Atributos

Declarações de listas de atributos identificam que elementos podem ter atributos, quantos e quais atributos eles podem ter, que valores os atributos podem suportar e se necessário qual será o valor padrão do atributo. Uma declaração de uma lista de atributos se parece com o seguinte exemplo:

```
<!ATTLIST livro nome ID #REQUIRED
               ano CDATA #IMPLIED
               estado-conservacao (bom | ruim) 'bom'>
```

Neste exemplo, o elemento *livro* possui dois atributos: nome, que é um *ID* (identificador) e é obrigatório (*#REQUIRED*), *ano* que é uma cadeia de caracteres (*CDATA*) e não é obrigatório (*#IMPLIED*) e *estado-conservacao* que pode ser ou *bom* ou *ruim* e por padrão é *bom* se nenhum valor é especificado.

Cada atributo em uma declaração pode ser constituído de três partes: um nome, um tipo e um valor padrão. É dada liberdade para selecionar o nome do atributo, sujeito algumas pequenas restrições, mas nomes não podem ser repetidos no mesmo elemento.

Existem seis tipos de atributos possíveis:

1. ***CDATA***

   Atributos do tipo *CDATA* são aqueles que possuem cadeias de caracteres em seus valores, ou seja, qualquer texto é permitido no valor do atributo. Não se deve confundir atributos *CDATA* com seções *CDATA*, eles não possuem relação alguma.

   Exemplo de uma declaração de um atributo do tipo CDATA:

   <!ATTLIST livro nome CDATA #IMPLIED>

2. ***ID***

   O valor de um atributo do tipo *ID* deve ser um nome, pois esse nome será usado como uma identificação de seu elemento. Todos os valores usados para os atributos *IDs* em um documento XML devem ser diferentes, para que os atributos *IDs* identifiquem univocamente elementos em um documento XML. Cada elemento pode ter um único atributo *ID*.

   No exemplo a seguir, o atributo *id* será o identificador *ID* do elemento *livro*:

   <!ATTLIST livro id ID #REQUIRED>

3. ***IDREF*** ou ***IDREFS***

   O valor de um atributo *IDREF* deve ser o valor de um único atributo *ID* de algum elemento no documento XML. Este atributo em um elemento será utilizado como um apontador para um outro elemento identificado pelo atributo *ID* contendo o mesmo valor do atributo *IDREF*.

   O valor de um atributo *IDREFS* pode conter valores *IDREF* múltiplos separados por espaços em branco.

   No exemplo a seguir, o atributo *ref* apontará para o identificador *ID* de um outro elemento:

   <!ATTLIST livro ref IDREF #IMPLIED>

4. ***ENTITY*** ou ***ENTITIES***

   O valor de um atributo *ENTITY* deve ser o nome de uma única entidade (veja a seguir sobre declarações de entidades).

   O valor de um atributo *ENTITIES* pode conter valores de múltiplas entidades separadas por espaços em branco.

   Esses atributos apontam para um dado externo do documento XML, por exemplo, para o conteúdo de um outro documento, que está sendo utilizado através de entidades no

documento XML.

No exemplo a seguir, o atributo *a* aponta para uma única entidade:

<!ATTLIST A a ENTITY #IMPLIED>

5. ***NMTOKEN*** ou ***NMTOKENS***

   Esses atributos de símbolos (*"tokens"*) são uma forma restrita do atributo de cadeia de caracteres (*CDATA*), ou seja, o valor de um atributo *NMTOKEN* continua sendo uma sequência de caracteres, porém um atributo *NMTOKEN* deve consistir de uma única palavra, mas não há restrições adicionais para essa palavra, e não é necessário estar associado com nenhum outro atributo ou declaração no DTD.

   O valor de um atributo *NMTOKENS* pode conter valores *NMTOKEN* múltiplos separados por espaços em branco.

   No exemplo a seguir, o atributo possui um token (símbolo) como valor:

   <!ATTLIST A a NMTOKEN #IMPLIED>

6. ***Lista de nomes***

   Com uma lista de nomes é possível especificar quais valores um atributo pode ter a partir de uma lista específica de nomes. Isto é freqüentemente chamado de tipo enumerado, pois cada um dos valores possíveis está explicitamente enumerado na declaração do atributo.

   Pelo exemplo, o elemento *documento* possui o atributo *tipo* que pode ser um dos seguintes valores: *livro*, *artigo* ou *revista*.

   <!ATTLIST documento tipo (livro | artigo | revista) #REQUIRED>

Os atributos podem ser opcionais, obrigatórios ou podem possuir um valor fixo. Atributos opcionais podem possuir um valor padrão, enquanto os atributos fixos devem possuir um padrão definido. Existem quatro possibilidades para os atributos:

1. ***#REQUIRED***

   O atributo deve possuir um valor explicitamente especificado em cada ocorrência do elemento no documento XML.

   No exemplo a seguir, todos os elementos *livro* do documento XML deverão possuir o atributo *ano*:

   <!ATTLIST livro ano CDATA #REQUIRED>

2. ***#IMPLIED***

   Neste caso o atributo não é obrigatório, e nenhum valor padrão é fornecido. Se um valor para este atributo não é especificado, o processador XML deve proceder normalmente sem esse atributo.

   No exemplo a seguir, os elementos *livro* não são obrigados a possuírem o atributo *ano*:

   <!ATTLIST livro ano CDATA #IMPLIED>

3. ***"valor"***

   Qualquer valor válido pode ser dado a um atributo como padrão. O valor do atributo não é requerido em cada elemento no documento, e se ele estiver presente será dado a ele o valor padrão.

   No exemplo a seguir, o valor padrão para o atributo *ano* será *2002*, mas este pode ser alterado no documento XML:

   <!ATTLIST livro ano CDATA "2002">

4. ***#FIXED "valor"***

   Uma declaração de atributo pode especificar que um atributo tem um valor fixo. Neste caso o atributo não é requerido, mas se ele ocorrer deve ter o valor especificado. Se não estiver presente será dado a ele o valor padrão. O uso de atributos fixos serve para associar uma semântica e um elemento. Por exemplo:

   <!ATTLIST livro ano CDATA #FIXED "2002">

O processador XML executa a normalização dos valores dos atributos, como por exemplo: as referências de caracteres que são substituídas pelos caracteres referenciados, referências a entidades que são resolvidas e espaços em branco que são normalizados.

2.2.4 Declarações de Entidades

Com as declarações de entidades é possível associar um nome com algum outro fragmento de conteúdo. Esse fragmento pode ser uma parte de um texto normal, uma parte de uma declaração DTD ou uma referência a um arquivo externo contendo texto ou dados binários.

Declarações de entidades são mostradas no exemplo da figura 2.5:

```
<!ENTITY Unicamp "Universidade Estadual de Campinas">
<!ENTITY noticia SYSTEM "/standard/noticia.xml">
<!ENTITY logo SYSTEM "/standard/logo.gif" NDATA GIF87A>
```

Figura 2.5: Exemplo de declarações de entidades em um DTD.

Existem três tipos de entidades:

1. ***Entidades Internas***

   As entidades internas associam um nome com uma cadeia de caracteres ou texto literal. A primeira entidade no exemplo da figura 2.5 é uma entidade interna. Usando *&Unicamp;* em qualquer lugar do documento XML será inserido a expressão literal *"Universidade Estadual de Campinas"* naquele local.

   Entidades internas permitem a definição de atalhos para textos freqüentemente digitados ou textos que se espera que sejam alterados, como o estado de revisão de um documento. Entidades internas podem incluir referências para outras entidades internas, mas não podem ser recursivas.

A especificação XML pré-define cinco entidades internas:

- ***<*** que produz o sinal de menor (<)
- ***>*** que produz o sinal de maior (>)
- ***&*** que produz o "E" comercial (&)
- ***'*** que produz um apóstrofo (')
- ***"*** que produz aspas (")

2. ***Entidades Externas***

As entidades externas associam um nome com o conteúdo de um outro arquivo. Entidades externas permitem a documento XML referenciar o conteúdo de um outro arquivo. As entidades externas podem conter texto ou dados binários. Se as entidades externas contêm texto, o conteúdo do arquivo externo é inserido no ponto de referência e analisado como parte integrante do documento referente. Dados binários não são analisados e podem somente ser referenciados por um atributo. Entidades externas que contenham dados binários são usados para referenciar figuras ou algum outro conteúdo "não-XML" no documento.

A segunda e a terceira entidade no exemplo da figura 2.5 são entidades externas.

O uso de *&noticia;* irá inserir o conteúdo do arquivo localizado por *"/standard/ noticia.xml"* no local da referência da entidade. O processador XML analisará o conteúdo deste arquivo como se ele ocorresse literalmente no local indicado pela entidade externa.

A entidade *logo* também é uma entidade externa, mas o seu conteúdo é binário. A entidade *logo* pode ser usada somente como o valor de um atributo *ENTITY* (ou *ENTITIES*). O processador XML passará esta informação para a aplicação, mas ele não tenta processar o conteúdo de *"/standard/logo.gif"*.

3. ***Entidades Parâmetro***

A entidade parâmetro somente pode ocorrer na declaração de um DTD. Uma declaração de uma entidade parâmetro é identificada por "% " (porcento e espaço) defronte ao seu nome na declaração. O sinal de porcento também é usado em referências para entidades parâmetro ao invés do "E" comercial (&). As referências à entidade parâmetro são imediatamente expandidas no DTD e seu texto de substituição é parte integrante da declaração, onde as demais referências a entidades não são expandidas. Entidades parâmetro não são reconhecidas no corpo de um documento XML, apenas em seu DTD.

Voltando às declarações de elementos, tomando o exemplo a seguir é possível perceber que esses dois elementos têm o mesmo modelo de conteúdo:

```
<!ELEMENT livro (autor+, titulo, ano)>
<!ELEMENT artigo (autor+, titulo, ano)>
```

Até o momento, esses dois elementos parecem ser os mesmos somente porque esses têm a mesma definição literal. A fim de tornar mais explícito o fato de que estes dois elementos são semanticamente iguais, é usada uma entidade parâmetro para definir seus modelos de conteúdo.

Há duas vantagens em se usar uma entidade parâmetro. Primeiramente, é possível dar um nome descritivo ao conteúdo, e segundo que também permite alterar o modelo de conteúdo somente em um local. Se for necessário atualizar as declarações do elemento, garantindo que elas sempre sejam as mesmas, basta realizar da seguinte forma como segue o exemplo:

```
<!ENTITY % conteudo-documento "autor+, titulo, ano">
<!ELEMENT livro (%conteudo-documento;)>
<!ELEMENT artigo (%conteudo-documento;)>
```

2.2.5 Declarações de Notação

Declarações de notação identificam tipos específicos de dados binários externos. Estas informações são passadas para a aplicação de processamento, que pode fazer o uso que quiser dessas informações. Um exemplo de uma declaração de notação:

<!NOTATION GIF87A SYSTEM "GIF">

2.3 DOM - Document Object Model

DOM (Document Object Model) [14] é uma API (Applications Programming Interface) independente de plataforma e linguagem que provê formas de acesso aos dados estruturados através da sua implementação em programas ou scripts, possibilitando ao desenvolvedor manipular as informações XML consistentemente, acessando e atualizando o conteúdo de documentos XML.

Um documento XML pode ser posteriormente processado e os resultados deste processamento pode ser incorporado de volta no documento apresentado. Em outras palavras, DOM descreve como analisadores de XML retornam a informação contida em um documento XML.

O DOM representa um documento XML como uma árvore onde os nós são objetos que representam elementos, texto e outras estruturas sintáticas como: comentários, declarações de marcação, instruções de processamento, etc.

Essa API é definida em vários níveis:

- **Nível 0**: Funções conhecidas das linguagens script dos browsers
- **Nível 1**: Funcionalidades para navegação em documentos e manipulações.
- **Nível 2**: Modelos de folhas de estilo, filtros, modelos de eventos, e suporte a namespaces[2] .
- **Nível 3**: Opções para carregar e salvar DTDs, schemas, e para visualização de documentos e status de formatação.

[2]Namespaces são coleções de nomes utilizados como prefixo para os nomes dos elementos e atributos de um documento XML, evitando confusões que possam surgir como nomes de elementos iguais mas que definem dados diferentes.

2.4 SAX - Simple API for XML

SAX (Simple Api for XML) [25] é uma API baseada em eventos. Eventos são reportados para as aplicações através de chamadas *"callback"*.

Utilizando SAX, o documento XML é lido por um parser, que identifica cada elemento no momento em que é encontrado, fazendo uma chamada para um método especificado pelo programador enquanto o documento é lido. O parser SAX passa apenas uma vez pelo documento fonte e o programador deve fazer tudo que necessita nesta única passagem.

A interface SAX pode ser usada por parsers Java. Um parser SAX é uma classe que implementa essa interface. Esse parser percorre os nós da estrutura em árvore dos documentos XML, chamando os métodos das classes definidas pelo usuário.

Uma das vantagens do SAX é requerer pouca memória não dependente do tamanho do documento XML, pois o documento XML inteiro não fica residente na memória, o que é interessante em situações onde há processamento de grandes documentos.

Os eventos gerados pelo parser SAX podem estar entre os seguintes tipos de eventos:

- Quando o **início** do documento é encontrado.
- Quando o **final** do documento é encontrado.
- Quando o **tag inícial** de um elemento é encontrado.
- Quando o **tag final** de um elemento é encontrado.
- Quando **dados (caracteres)** são encontrados.
- Quando uma **instrução de processamento** é encontrada.

2.4.1 Características do SAX

Apesar das vantagens de pouca necessidade de memória e de processamento rápido, a análise realizada pelo parser SAX é seqüencial permitindo em cada momento acesso a uma única parte do documento: não é possível o acesso a dois elementos simultaneamente, e

também não se pode acessar um elemento e o seu conteúdo ao mesmo tempo. O acesso aleatório não é permitido e o programador deve controlar a posição no documento, como também não é possível o acesso a partes dos documentos que ainda não foram analisadas.

A interface SAX é de maior utilidade em aplicações no servidor, não sendo apropriada em aplicações no cliente onde se pretenda alterar/criar interativamente documentos XML.

2.4.2 Manipuladores SAX

Alguns dos principais manipuladores utilizados na interface SAX:

- *ErrorHandler*: Tratamento de erros ocorridos durante a análise sintática.
- *DTDHandler*: Tratamento de notações e entidades não analisadas.
- *EntityResolver*: Tratamento de entidades externas.
- *LexicalHandler*: Tratamento de comentários, definição de DTDs ou de entidades que existam no documento inicial e se pretende manter no documento final.
- *Locator*: Acesso à localização no documento fonte onde ocorreu um dado evento.
- *ContentHandler*: Recebe e manipula a informação acerca dos elementos analisados, dos seus atributos e do contéudo CDATA. É o mais usado. Alguns métodos do *ContentHandler*:
  - *setDocumentLocator(l)*.
  - *startDocument()* e *endDocument()*.
  - *startElement(n,as)* e *endElement(n)*.
  - *characters(c)*.
  - *processingInstruction(t,d)*.

2.5 DSSSL - Document Style Semantics and Specification Language

DSSSL (Document Style Semantics and Specification Language) [22] é um padrão ISO (ISO/IEC 10179:1996) que apresenta recursos para transformação e formatação de documentos a fim de padronizar o estilo da sintaxe e da semântica do *layout* de documentos SGML e XML ou fragmentos desses documentos. Todas as especificações de estilo em DSSSL descrevem o resultado da formatação final. DSSSL permite especificar como dados serão apresentados nos dispositivos de saída, sem criar o formato desses dados. Dos trabalhos com DSSSL surgiu a linguagem XSL, suportada pela W3C.

DSSSL consiste de dois componentes principais: uma linguagem de transformação e uma linguagem de estilo. A linguagem de transformação é usada para especificar as transformações estruturais em arquivos SGML ou XML; também pode ser usada para especificar a união de dois ou mais documentos, para a geração de índices e outras operações.

Com uma linguagem de estilo é possível identificar um número de possibilidades que por uma ou outra razão podem ser consideradas opcionais. Os desenvolvedores da DSSSL destinaram certas características como opcionais e criou um Núcleo de Linguagem de Consulta e um Núcleo de Linguagem de Expressão para deixar o mais limitado possível sua implementação. Não foi definido um conjunto particular de componentes da linguagem de estilo a partir do padrão propriamente dito, mas deixaram esse trabalho para organizações industriais e de padronização.

Um exemplo do uso da DSSSL na aplicação do modelo será visto na seção 4.2.1.

2.6 XSL - Extensible Style Language

XSL (Extensible Style Language) [2] [26] é uma linguagem de *"folhas de estilo"* destinadas à visualização de uma informação textual. Uma folha de estilo separa o conteúdo e a lógica estrutural de um documento de sua apresentação (figura 2.6), assim como CSS (Cascading Style Sheets) [5] o atual padrão para definir a apresentação de documentos HTML e DSSSL (Document Style Semantics and Specification Language.) [22], que também utilizam na implementação do modelo.

XSL consiste de duas partes:

1. *Transformações XSL (XSLT - XSL Transformation)*: utilizadas para transformar um documento XML em outro tipo de documento de marcação, por exemplo documentos HTML, XHTML ou XML possuindo diferentes elementos do documento de origem;

2. *Formatação de Objetos XSL (XSL-FO - XSL Formatting Objects)*: utilizada para definir objetos de formatação para apresentação gráfica de um documento XML, por exemplo, documentos PDF e PS.

Uma folha de estilo XSLT é um documento seguindo o padrão XML como mostra a figura 2.7.

XSL-FO é uma linguagem XML para especificar uma formatação de nível mais baixo e de forma mais detalhada do que é possível com HTML + CSS.

XSL é especialmente utilizada através da sua linguagem de transformação XSLT. É com este tipo de aplicação em mente que a XSL está implementada nos recentes browsers Web. Uma folha de estilo XSLT é declarativa e mais expressiva que uma folha de estilo CSS.

XSLT é, na verdade, uma linguagem de programação declarativa com recursos semelhantes aos da linguagem SQL.

Vimos que documentos XML são hierárquicos, isto é, podem ser vistos como uma árvore onde cada nível possui vários elementos. XSLT define transformações que podem abranger um elemento ou até vários níveis. A aplicação XSLT percorre o documento, verificando para cada nível se existe uma transformação associada. Caso exista, ela é aplicada. A escolha dos

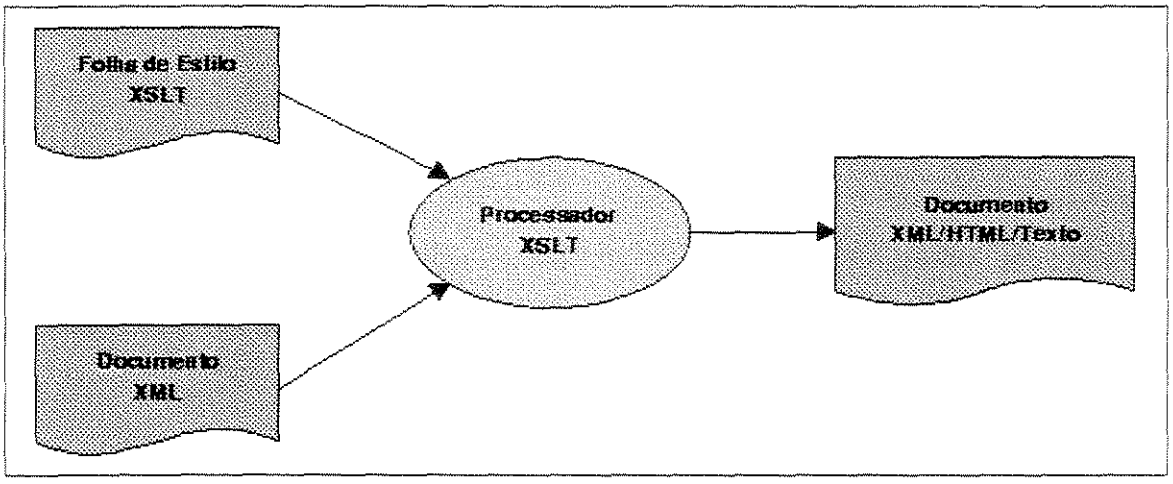

Figura 2.6: Idéia básica de XSLT.

elementos a transformar é feita através de operadores de seleção e de associações aos nomes dos elementos.

As transformações XSLT podem ser feitas pelo cliente (por exemplo, Internet Explorer 6), ou pelo servidor (por exemplo, Apache Xalan); elas podem ser pré-processadas ou geradas dinamicamente. XSLT vem sendo amplamente implementada, ao contrário de XSL-FO.

2.6.1 Modelo de Processamento

O modelo de processamento de uma folha de estilo XSLT [26] consiste em um número de modelo de regras. Para um documento XML como entrada, o resultado é obtido da seguinte maneira:

- A árvore do documento fonte é processada a partir do elemento raiz;
- Cada nó é processado da seguinte forma:
  - é feita uma busca de uma regra do modelo que combina com o nó em questão;
  - o modelo é instanciado (criando uma parte do resultado + continuação do processamento recursivamente).
- pelo processamento de cada nó é gerada uma lista ordenada de nós e é feita uma concatenação dos resultados.

2.6.2 Estrutura de um Folha de Estilo XSLT

Exemplo:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
      xmlns:=". . .">
      .
      .
      <xsl:template match="nome-elemento">
            <xsl:apply-templates/>          -> Regras do modelo.
      </xsl:template>
      .
      .          -> Outras definições de elementos.
</xsl:stylesheet>
```

Figura 2.7: Exemplo de folha de estilo XSLT.

Um documento XML para ser visualizado em browsers que contenham um processador XSLT, referencia uma folha de estilo usando uma instrução de processamento como a seguinte:

```
<?xml-stylesheet type="text/xsl" href="estilo.xsl"?>
```

Possibilidades do Modelo XSLT

Há muitos tipos diferentes de construções de modelos XSLT, que podem ser divididas em:

- *Processamento recursivo*:
  - *<xsl:apply-templates select="node-set" .../>*, aplica o modelo em nós selecionados.
  - *<xsl:call-template name="..."/>*, chamada de um modelo através do nome.
  - *<xsl:for-each select="node-set"> template </...>*, associa o modelo para cada nó selecionado.
- *Computação de fragmentos do resultado*:
  - *<xsl:element name="..." namespace="..."> ... </...>*, gera um elemento com nome, atributos e conteúdo fornecido.
  - *<xsl:value-of select="..."/>*, gera caracteres ou valores de atributos.

- *<xsl:processing-instruction name="...">* ... *</...>*, gera uma instrução de processamento.

- *Processamento condicional*:
  - *<xsl:if test="expression">* ... *</...>*, aplica o modelo se a expressão é verdadeira.
  - *<xsl:choose>*
    <xsl:when test="expression"> ... *</...>*
    <xsl:otherwise> ... *</...>*
    </...>,
    testa as condições e aplica o modelo da primeira expressão verdadeira.

- *Numeração*: Numeração automática de seções, lista de itens, rodapés, etc.
  - *<xsl:number value="expression"* -> por exemplo: *footnote.*
    format="..." -> padrão: 1.
    level="..." -> opções: *any/single/multiple.*
    count="..." -> O que será numerado.
    from="..." -> Inicio da numeração.
    grouping-size="..."/>.

- *Variáveis e parâmetros*: Reutilizam resultados já computados e parametrização de modelos. Obedecem a regras de escopo estático, podem ser valores como strings, números, conjunto de nós; são declarativas não podendo ser atualizadas e podem ser globais ou locais ao modelo.
  - *<xsl:variable name="..." select="expression"/>*, declaração de variável com a seleção de um modelo.
  - *<xsl:variable name="...">* *template* *</..>*, declaração de variável com o modelo instanciado diretamente em sua declaração.

Um Exemplo mais Detalhado do Uso de XSLT

A folha de estilo XSLT da figura 2.8 transforma um cartão de negócios no formato XML representado pela figura 2.9 em um documento final XHTML visualizado na figura 2.10.

Uma folha de estilo CSS é muito limitada comparada com XSLT. Em CSS os atributos não são exibidos, as informações não podem ser reorganizadas e nenhum tipo de computação é possível (como expressões condicionais, loops, etc).

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
      xmlns:="http://www.w3.org/1999/xhtml">
   <xsl:template match="card">
      <html>
         <head> <title><xsl:value-of select="name/text()" /></title> </head>
         <body bgcolor="#ffffff">
            <table border="3">
               <tr> <td>
                     <xsl:apply-templates select="name" /><br/>
                     <xsl:apply-templates select="title" /><p/>
                     <tt><xsl:apply-templates select="email" /></tt><br/>
                     <xsl:if test="phone">
                        Phone: <xsl:apply-templates select="phone" /><br/>
                     </xsl:if>
               </td> <td>
                     <xsl:if test="logo"> <img src="logo/@url" /><br/> </xsl:if>
               </td> </tr>
            </table>
         </body>
      </html>
   </xsl:template>
   <xsl:template match="/"> <xsl:value-of select="text()" /> </xsl:template>
</xsl:stylesheet>
```

Figura 2.8: Exemplo Prático de folha de estilo XSLT.

2.6.3 Formatação de Objetos XSL (XSL-FO)

XSL-FO [21] é a segunda metade da linguagem de estilo XSL. XSL-FO é uma aplicação XML que descreve como as páginas serão apresentadas a um leitor. Uma folha de estilo usa a linguagem de transformação XSL para transformar um documento XML em um vocabulário semântico dentro de um novo documento XML que utiliza o vocabulário de apresentação XSL-FO. Os browsers Web ainda não exibem diretamente documentos XML que utilizam XSL-FO, como já é possível com XSLT; esse processo necessita de um passo adicional gerando um documento em um outro formato como PDF (Portable Document Format) ou PS (PostScript).

```
<card>
      <name>John Doe</name>
      <tilte>CEO, Widget Inc.</tilte>
      <email>john.doe@widget.com</email>
      <phone>(202) 555-1414</phone>
      <logo url="widget.gif" />
</card>
```

Figura 2.9: Documento XML para a Transformação XSLT.

Figura 2.10: Documento XHTML resultado da Transformação XSLT.

XSL-FO fornece um controle exato e detalhado do *layout* dos documentos finais, provendo um modelo visual mais sofisticado que a utilização de HTML+CSS, incluindo por exemplo cabeçalhos, rodapés e numeração de páginas. Enquanto CSS é utilizado especificamente para a Web, XSL-FO foi desenvolvido para uma utilização mais ampla.

É possível com folhas de estilos XSL utilizando formatação de objetos (XSL-FO) gerar um livro inteiro para impressão. Utilizando as transformações XSL (XSLT) é possível transformar o mesmo documento XML em um documento para Web.

O exemplo da figura 2.11 ilustra um uso simples de XSL-FO, iniciando com algumas definições XSL-FO e XML (já que também é um documento XML). Nesse exemplo o elemento **fo:layout-master-set** define o *layout* da página formatando o tamanho das margens, cabeçalho e rodapé; em seguida as páginas do documento são agrupadas pelo elemento **fo:page-sequence**, onde o conteúdo do documento é agrupado em blocos (parágrafos) pelo

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
        <fo:simple-page-master master-name="my-page">
            <fo:region-body margin="1in"/>
        </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-name="my-page">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-family="Times" font-size="14pt">
                <fo:inline font-weight="bold">Hello</fo:inline>, world!
            </fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

Figura 2.11: Um exemplo simples de XSL-FO.

elemento **fo:block**; ainda é possível associar apenas algumas propriedades em fragmentos dos blocos através do elemento **fo:inline**.

Um exemplo do uso da XSL na aplicação do modelo será visto na seção 4.2.2.

Capítulo 3

Trabalhos Relacionados

3.1 DocBook

O DocBook [35] é um sistema para escrever documentos estruturados usando a linguagem SGML e mais recentemente XML. DocBook disponibiliza um conjunto de elementos (a partir de um DTD) para marcar determinadas características num texto, descrevendo-o de modo que uma ferramenta externa possa convertê-lo para diversos formatos. O conjunto de elementos DocBook pode ser usado na elaboração de documentos técnicos, como por exemplo livros e artigos. DocBook é um exemplo de linguagem de marcação definida originalmente em SGML e recentemente em XML.

Para a publicação de documentos eletrônicos um DTD foi desenvolvido inicialmente pela IBM. Hoje se encontra bem difundido com muitas publicações utilizando sua concepção e parametrização. Docbook é utilizado pela IBM e pela O'Reilly na maior parte das suas publicações. Há também um grande suporte para Docbook junto às ferramentas de SGML e XML.

Via arquivos XML, DTD e folhas de estilo DSSSL ou XSL, Docbook permite gerar docu-

mentos HTML, XML, RTF e LaTeX, entre outros formatos, por exemplo através de ferramentas de processamento como "Jade" [11]. Jade é um processador de documentos markup XML e SGML utilizando folhas de estilo em DSSSL; possibilita converter arquivos padronizados pelo DocBook nos formatos citados anteriormente utilizando folhas de estilo em DSSSL escritas especificamente para DocBook. Outro exemplo de processador para XSL é o Saxon (processador XSLT escrito em Java) [23], que utiliza folhas de estilo XSL para processar documentos XML.

A idéia do DocBook é construir um documento apenas uma vez (no formato XML ou SGML) e atingir o maior público possível utilizando diferentes formatos nas publicações.

O DocBook possui mais de 300 elementos, cada qual com diversos atributos que podem tomar diversos valores, fixos ou definidos pelo autor do documento.

É possível inserir figuras em formatos específicos para os tipos de mídia nas quais o documento será publicado. Por exemplo, se usarmos o formato LaTeX será possível gerar imagens em PostScript (PS ou EPS - Encapsulated PostScript), através do elemento DocBook *<figure>*. Para publicações on-line é necessário um formato acessível por um navegador Web, como por exemplo, JPG, GIF ou PNG. Também é possível a utilização de tabelas através do elemento *<table>* e seus demais sub-elementos (linhas, colunas, título, etc), já que muitas informações são melhor representadas quando formatadas em tabelas. A criação de um índice remissivo em DocBook é realizada através de marcações no texto usando os elementos *<indexterm>*, *<primary>*, *<secondary>* e *<tertiary>*.

Para cada tipo de documento pode ser criada uma estrutura particular de cabeçalho, conteúdo (textos, imagens, tabelas, etc) e rodapé.

A tabela 3.1 mostra exemplos simples de instruções para escrever um documento usando DocBook:

Função	Elemento docbook	Resultado final
Endereço de correio eletrônico	<email>endereco@domínio</email>	<endereço@domínio>
Referência bibliográfica no texto	<citation>referência</citation>	[referência]
Enfatizar o texto (itálico)	<emphasis>texto</emphasis>	*texto*
Listas com marcadores	<itemizedlist> <listitem> <para>item</para> </listitem> <listitem> <para>item</para> </listitem> </itemizedlist>	• item • item
Lista numerada	<orderedlist> <listitem> <para>item</para> </listitem> <listitem> <para>item</para> </listitem> </orderedlist>	1. item 2. item

Tabela 3.1: Exemplo de elementos DocBook.

3.1.1 Inserindo Figuras

Uma maneira de inserir figuras em documentos DocBook é através do uso do atributo *fileref*, ilustrado na figura 3.1 a seguir. Usualmente geram-se figuras em JPG ou em PostScript (PS ou EPS).

```
<figure>
    <title>Título da Figura</title>
    <graphic fileref="imagens/arquivo"></graphic>
</figure>
```

Figura 3.1: Inserindo uma figura em um documento DocBook.

Substituindo-se *<figure>* por *<informalfigure>* elimina-se a necessidade de inserir um título para a figura.

3.1.2 Exemplo de Documento DocBook

As sub-seções seguintes darão exemplos de cabeçalhos e estruturas válidas para *artigos* e *livros*, com a intenção de ilustrar genericamente o que pode ser feito com DocBook.

Exemplo de Artigo

A figura 3.2 mostra um exemplo simples para a criação de um artigo DocBook.

Exemplo de Capítulo

Um capítulo começa sempre com o comando *chapter*; para criar uma seção utilize o comando *section*. Este comando é seguido pelo comando *title* que define o título do capítulo em questão, na figura 3.3.

```
<article class="whitepaper" id="usando-docbook" lang="pt-br">
  <artheader> <title>Exemplo de Artigo DocBook</title>
    <author>
      <firstname>Jorge</firstname> <othername>Luiz</othername>
      <surname>Filho</surname>
      <affiliation>
        <orgname><ulink url="http://www.ic.unicamp.br">
          Instituto de Computação - UNICAMP</ulink></orgname>
        <orgdiv>Setor de Publicações</orgdiv>
        <address><email>jorge@ic.unicamp.br</email></address>
      </affiliation>
    </author>
    <revhistory>
      <revision>
        <revnumber>1.0</revnumber> <date>27 de julho de 2002</date>
        <authorinitials>jorge</authorinitials>
        <revremark>Versão inicial.</revremark>
      </revision>
    </revhistory>
    <legalnotice>
      <para>Este documento pode ser livremente distribuído e traduzido. Ele está
      liberado sob a GDPL - GNU Documentation Public License.</para>
    </legalnotice>
    <keywordset>
      <keyword>DocBook</keyword> <keyword>DTD</keyword>
      <keyword>XML</keyword> <keyword>catálogos</keyword>
      <keyword>documentos</keyword> <keyword>Publicações</keyword>
    </keywordset>
  </artheader>
```

Figura 3.2: Exemplo de um artigo.

```
<chapter id="capitulo1">
      <title>Título do Capítulo</title>
      <para>Conteúdo do capítulo.</para>
</chapter>
```

Figura 3.3: Exemplo de capítulo.

3.1.3 Extensões dos Elementos DocBook para Aplicação no Catálogo de Cursos da Unicamp

Um dos primeiros passos deste trabalho foi a análise do tipo de documento que se quer desenvolver e conseqüentemente, decidir pela escolha de um DTD existente para essa classe se possível, ou pela criação de um novo DTD que descreva as características próprias da aplicação. Como aplicação prática desse trabalho foram analisados os tipos de documentos utilizados pela Unicamp na construção dos catálogos de cursos de graduação e de pós-graduação, tanto para a utilização dos documentos para impressão quanto para disponibilização na Web.

Analisando os elementos disponíveis no DocBook para o desenvolvimento de artigos e livros foi necessário estender alguns elementos do DocBook com o intuito de melhor representar as características dos catálogos de cursos e a definição de um novo DTD para a aplicação. Alguns dos elementos DocBook usados como base para a aplicação estão na tabela 3.4

Outros elementos foram criados para representar características próprias dos catálogos, como elementos para informações sobre faculdades, institutos, departamentos, comissões, diretores, corpo docente, cursos e disciplinas. O DTD completo da aplicação está descrito no capítulo 5.

Elementos DocBook	Elementos para os Catálogos
<address> utilizado como endereço	*<endereco>* também utilizado como endereço
<author> nome do autor de um livro ou artigo	*<nome>* utilizado como nome de reitor, diretores, professores, departamentos e comissões, disciplinas, cursos, etc
<bibliography> bibliografia de um livro ou artigo	*<bibliografia>* bibliografia utilizada por uma disciplina
<para> parágrafo dentro de um texto	*<paragrafo>* parágrafo dentro de um texto
<title> título de uma seção de um documento	*<titulo>* título de algumas partes dos catálogos
<subtitle> sub-título de uma seção de um documento	*<subtitulo>* sub-título de algumas partes dos catálogos
<year> ano de publicação de um documento	*<ano>* descrição do ano em várias partes dos catálogos
<emphasis> enfatiza parte do texto	*<negrito>* enfatiza parte do texto
<figure> utilizado para inserir uma imagem	*<imagem>* utilizado para inserir uma imagem

Figura 3.4: Exemplo de algumas extensões DocBook para a DAC.

3.2 Armazenamento e Consultas de Documentos XML

Com o crescimento da tecnologia XML para o intercâmbio e representação de dados na Web surgiram problemas relacionados ao armazenamento, atualização e recuperação da informação no formato XML. Uma boa opção para solucionar esses problemas seria o uso de sistemas de banco de dados, pois esses estão em um alto grau de desenvolvimento oferecendo vantagens como uma grande confiabilidade, escalabilidade e desempenho.

O armazenamento de documentos XML e a possibilidade de executar consultas sobre esses dados podem ser realizados das seguintes maneiras: criando um sistema de banco de dados específico armazenando os documentos XML, ou então usando um sistema de banco de dado orientado a objetos suportando bem as estruturas aninhadas dos documentos XML, ou ainda utilizando um sistema de banco de dados relacional através do mapeamento de documentos XML em tabelas do esquema relacional e a transformação de consultas XML em consultas SQL (Structured Query Language).

Analisando as possibilidades apresentadas, um sistema de banco de dados específico seria o ideal, mas levarão um bom tempo para adquirir um grau de desenvolvimento capaz de suportar a manipulação de grandes quantidades de dados de forma eficiente. Já os sistemas de banco de dados orientado a objetos não estão suficientemente desenvolvidos para suportar consultas em grandes bancos de dados. A outra alternativa seria o uso de sistemas de bancos de dados relacionais que se encontram suficientemente desenvolvidos para suportar o armazenamento e a execução de consultas em grandes bancos de dados, além da possibilidade da construção de aplicações sobre sistemas de banco de dados relacionais que manipule dados em XML e dados tradicionais estruturados sem muito esforço. Mas os requisitos para o processamento dos dados XML são bem diferentes dos dados tradicionais armazenados pelos sistemas de banco de dados relacionais, dificultando essa adaptação.

Sistemas de banco de dados relacionais são maduros e escaláveis sendo uma alternativa recente bastante pesquisada para a integração com o padrão XML: estudo de modelos e algoritmos para extrair esquemas dos documentos XML, integração de XML com sistemas de

banco de dados relacionais, criação de métodos de transformação de consultas para documentos XML em consultas SQL e algoritmos para reconstrução de documentos XML a partir de banco de dados relacionais.

Este é um tema complexo e apenas um sumário será apresentado a seguir; ele está sendo tratado com maior profundidade numa outra dissertação em progresso.

3.2.1 Mapeamento de Documentos XML para Bancos de Dados Relacionais

Para o mapeamento de documentos XML para bancos de dados relacionais, primeiramente o método mais simples seria armazenar cada documento XML em uma tabela do banco de dados relacional, mas a desvantagem estaria no comprometimento do desempenho das consultas e atualizações.

Um outro método de mapeamento seria a utilização de um grafo que representasse um documento XML e gerar um esquema relacional que permita armazenar qualquer estrutura de um grafo. Um documento XML pode ser representado por um grafo direcionado modelando seus elementos em vértices, seus atributos e sub-elementos em arestas e seus valores nas folhas do grafo. As arestas do grafo são rotuladas com os nomes dos atributos ou sub-elementos que representam e cada vértice possui um identificador. Um esquema relacional simples armazena os atributos em uma tabela com os campos *origem* indicando qual o vértice de origem, *nome* contendo o nome do atributo ou sub-elemento e o campo *valor* representando o valor final ou uma referência para um outro sub-elemento. Há uma simplificação na representação dos dados através de grafos podendo perder informação durante o processo, pois não há diferenciação entre atributos sub-elementos e suas referências. Os documentos XML reconstruídos a partir desses bancos de dados relacionais poderão ser diferentes dos originais.

Outra alternativa seria o armazenamento de documentos XML de acordo com um esquema relacional mapeado através do DTD. Um método simples seria criar esquemas relacionais para cada elemento do DTD contendo todos os seus sub-elementos e atributos. Essa

alternativa é que melhor explora a funcionalidades de um SGBD através de mecanismos de consulta, otimização e controle de concorrência. Mas nos documentos XML existem redundância, atributos multivalorados e recursão, o que dificulta o seu mapeamento. E nem todo documento XML possui um DTD para ser mapeado.

3.2.2 Transformação de Consultas XML em consultas SQL

Tendo um documento XML armazenado em um banco de dados relacional seria necessário a execução de consultas e a atualização desses dados. Existem várias linguagens para XML, como por exemplo, XML-QL: *A Query Language for XML* [12], *XQL: XML Query Language* [30] e *XQuery 1.0: An XML Query Language* [4] sendo a linguagem mais cotada para ser adotada como a linguagem padrão, mas ainda não foi definida pela W3C.

Para utilizar o poder dos sistemas de bancos de dados relacionais e executar consultas sobre a informação XML armazenada, as consultas definidas nas linguagens XML precisam ser transformadas em consultas SQL. Existem projetos relacionados com a integração de XML e bancos de dados relacionais que abordam a transformação de consultas XML em consultas SQL. Como ocorre no projeto SilkRoute [16]onde é utilizada uma linguagem especial chamada RXL (Relational to XML Transformation Language), que possui as cláusulas *from* e *where* semelhantes ao SQL e a cláusula *construct* que constrói a informação XML.

Uma outra solução [10], parecida com a citada anteriormente, transforma consultas definidas na linguagem de consulta XML-QL em consultas SQL.

Apesar de existirem boas soluções, a transformação de consultas XML em SQL apresenta algumas limitações devido às grandes diferenças entre as linguagens de consultas XML e a linguagem SQL, e nem tudo pode ser transformado.

3.2.3 Reconstrução de Documentos XML a partir de Banco de Dados Relacionais

Na reconstrução de documentos XML a partir de banco de dados relacionais existem duas possibilidades.

A primeira alternativa seria a utilização de linguagem de consultas XML, que seriam transformadas em consultas SQL, para selecionar a informação desejada e construir um documento XML. Como ocorre no projeto SilkRoute [16] citado anteriormente, a linguagem RXL é responsável pela criação de tags no documento XML resultante. Mas é necessário conhecer um linguagem de consulta XML específica para ser realizada.

A outra possibilidade consiste na utilização de consultas SQL e o mapeamento dos resultados tabulares retornados para o padrão XML que possui uma estrutura mais complexa. A maior dificuldade dessa alternativa é a necessidade de conhecer o esquema do banco de dados relacional para realizar a consulta SQL, o que dificulta a sua realização.

Capítulo 4

Aplicação de XML aos Catálogos de Cursos

Nesse capítulo será apresentado o processo para automatizar a troca de informação entre as unidades responsáveis pelos cursos oferecidos pela Unicamp e a Diretoria Acadêmica (DAC), responsável pela publicação dos catálogos e sua disponibilização na Web.

Para exemplificar o processo de criação da informação XML e a sua conversão para outros formatos de documentos, será utilizada a descrição de uma disciplina de graduação da Unicamp. Foram utilizados os conceitos abordados anteriormente e alguns conceitos serão detalhados neste capítulo, como os documentos de estilo utilizados na apresentação da informação XML.

4.1 Exemplo de XML e seu DTD para o Catálogo de Cursos

Documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE disciplina SYSTEM "Disciplina.dtd">
<disciplina>
   <codigo>MC202</codigo>
   <nome>Estruturas de Dados</nome>
   <pre-requisito><codigo>MC102</codigo></pre-requisito>
   <ementa>Representação e manipulação de informações. Eficiência. Estruturas
           básicas: listas e árvores e suas generalizações. Tipos abstratos
           de dados e objetos. Desenvolvimento, implementação e testes
           de programas em aplicações particulares.
   </ementa>
</disciplina>
```

Figura 4.1: Exemplo de um documento XML representando a ementa de uma disciplina.

O exemplo da figura 4.1 apresenta a descrição dos elementos e seus relacionamentos em um documento XML referente a uma disciplina dos catálogos de cursos.

O documento começa com uma instrução de processamento: *<?xml ...?>*. Esta é a declaração XML. Embora não seja obrigatória, a sua presença explícita identifica o documento como um documento XML e indica a versão da XML com a qual ele foi escrito.

O atributo existente na instrução de processamento *encoding="ISO-8859-1"* informa que o documento está usando um conjunto de caracteres propício para o Português, permitindo a acentuação das palavras.

Na segunda linha, o elemento *DOCTYPE* indica qual documento DTD será utilizado.

A ementa de uma disciplina feita na figura 4.1 utiliza os seguintes elementos:

- *codigo*: representa o código da disciplina;
- *nome*: representa o nome da disciplina;
- *pre-requisito*: representa os códigos das disciplinas que são pré-requisito para a disciplina em questão;
- *ementa*: representa os assuntos que serão abordados pela disciplina.

Documento DTD:

```
<!ELEMENT disciplina (codigo, nome, pre-requisito?, ementa)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT pre-requisito (codigo+)>
<!ELEMENT ementa (#PCDATA)>
```

Figura 4.2: Exemplo do DTD do documento XML.

O DTD da figura 4.2 descreve a estrutura válida para um documento XML de uma disciplina do catálogo de cursos: o elemento *disciplina* deve conter os elementos *codigo*, *nome* e *ementa*; já o elemento *pre-requisito* é opcional, podendo ou não estar presente. O elemento *pre-requisito* contém um ou mais elementos *codigo*. Os demais elementos com a definição *#PCDATA* poderão conter texto entre seus *tags*.

O Diagrama da figura 4.3 apresenta a estrutura hierárquica dos elementos declarados no DTD da figura 4.2

A publicação dos catálogos graduação e pós-graduação contêm uma grande quantidade de informação, e muitas vezes a informação se repete dentro de um catálogo e pode se encontrar

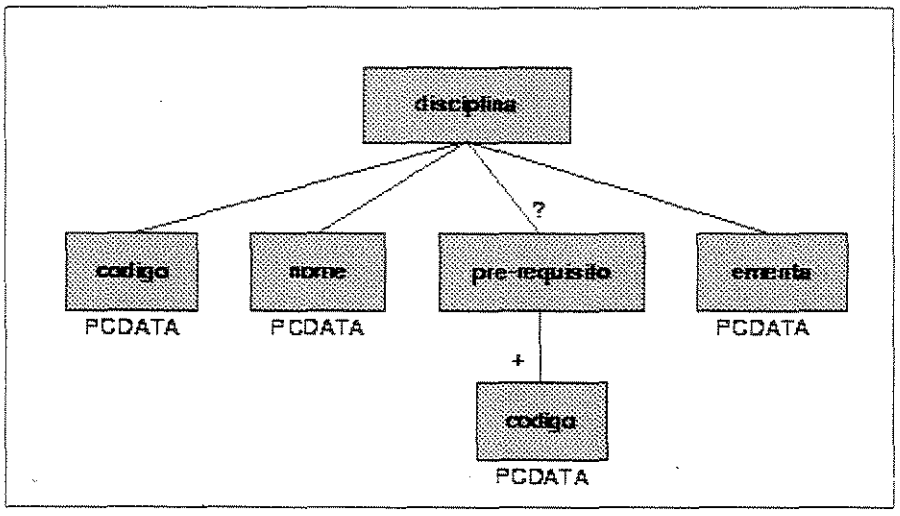

Figura 4.3: Diagrama de hierarquia do DTD da figura 4.2.

em ambos os catálogos. Para evitar esta redundância de informação, a tecnologia XML disponibiliza recursos de identificação da informação através de um atributo definido em seu DTD, o atributo *"ID"* que identifica um elemento dentro do documento XML e o valor desse atributo deve ser único.

Definição no documento DTD: *<!ATTLIST nome ident ID #REQUIRED>*.

Utilização do atributo ID no documento XML: *<nome ident="ED">Estrutura de Dados </nome>*.

Para partes do documento que reutilizarão a informação identificada, é definido um outro atributo chamado *"IDREF"* utilizado como referência para um elemento, e o valor desse atributo deve ser igual a um identificador *ID* de um elemento, evitando assim a reescrita da informação em várias partes do documento XML.

Definição no documento DTD: *<!ATTLIST nome ref-ident IDREF #REQUIRED>*.

Utilização do atributo IDREF no documento XML: *<nome ref-ident="ED"></nome>*.

Dessa maneira é realizada a definição apenas uma vez do elemento *nome* de uma disciplina, em sua primeira utilização; nas demais necessidades de sua descrição é feita apenas uma referência à primeira definição, repetindo automaticamente a informação do elemento.

A seguir serão apresentados as formas de criação dos documentos XML e a conversão para os diferentes formatos como mostra a figura 4.4.

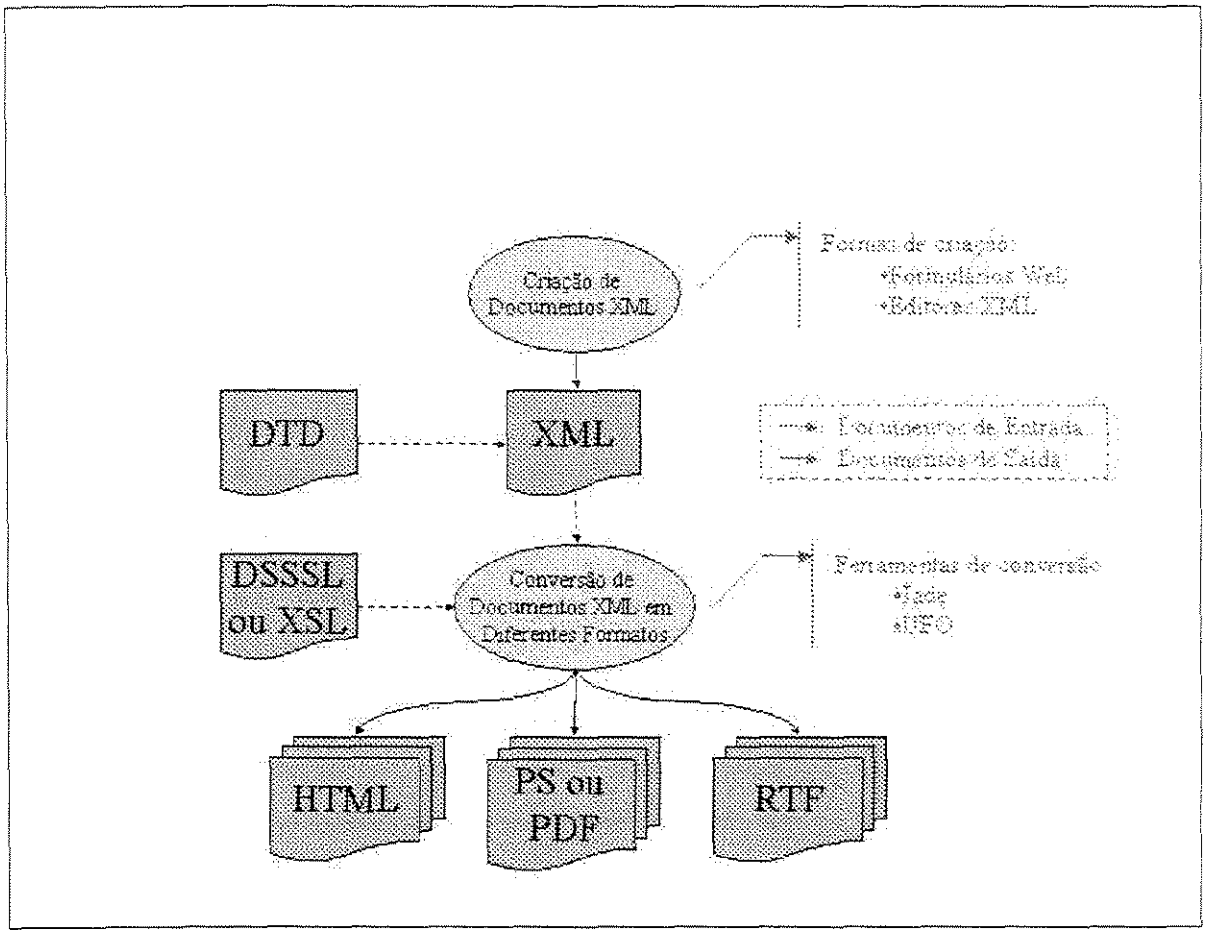

Figura 4.4: Diagrama do Modelo de Intercâmbio.

4.2 Criação de Documentos XML

Inicialmente será necessária a geração dos dados dentro do padrão XML para posteriormente ser realizada a troca de informação entre as partes envolvidas na aplicação, e também a publicação e disponibilização dessa informação em diferentes formatos para os usuários finais da aplicação.

A geração da informação no formato XML não pode ocorrer de forma aleatória sem alguma organização dos dados: é preciso haver uma certa ordenação seguindo regras para formatar os dados e detalhadas anteriormente no capítulo 2.

Essas regras são especificadas através de um DTD, onde é descrita a gramática que o documento XML deve seguir.

A criação de documentos XML merece uma atenção especial, já que na maioria dos casos envolverá uma grande quantidade de informação, gerando estruturas hierárquicas extensas e complexas e consequentemente dificultando a criação de documentos obedecendo às regras definidas para a aplicação.

Existem ferramentas que podem auxiliar na criação de documentos no formato XML para uma dada aplicação, como editores de documentos XML ou desenvolvimento de formulários Web que receberão as informações requeridas para a criação do documento XML com as características da aplicação.

4.2.1 Editores XML

Os editores XML basicamente auxiliam na separação do que é tag do que é conteúdo (informação), necessitando um conhecimento dos usuários sobre a linguagem XML para a utilização correta dos editores. Alguns editores XML também auxiliam na inserção de tags a partir de elementos definidos em um DTD para a aplicação.

Foram pesquisados vários editores XML priorizando a sua disponibilidade de uso livre ou comercial, plataforma utilizada e facilidade na manipulação da ferramenta. Alguns editores XML também fornecem auxílio para a criação de DTDs.

A tabela 4.1 contém uma lista de editores XML e algumas de suas características, ordenada por funcionalidade e praticidade:

Nesta apresentação será utilizado como exemplo uma disciplina do catálogo de graduação atendendo às regras definidas pela aplicação do catálogo, com o objetivo de criar um documento XML ilustrado pela figura 4.1.

Editor XML	*Disponibilidade*	*Plataforma*	*Descrição*
Epcedit	Open source	Windows/ Linux	Editor visual, auxilia na inserção de tags e na separação dos elementos e conteúdo com cores diferentes. Analisa o documento a partir de um DTD. Exibe estrutura hierárquica.
EditML Pro	Comercial (Trial)	Windows	Editor visual, que também auxilia na criação de DTD. Analisa o documento a partir de um DTD. Exibe estrutura hierárquica.
Athens XML Editor	Comercial (Trial)	Windows	Auxilia na separação dos elementos, atributos e conteúdo, através de cores diferentes. Exibe estrutura hierárquica.
Peter's XML Editor	Freeware	Windows	Auxilia na separação dos elementos, atributos e conteúdo, através de cores diferentes. Exibe estrutura hierárquica.
Cooktop	Freeware	Windows	Auxilia na separação dos elementos, atributos e conteúdo, através de cores diferentes. Auxilia na criação de DTD.

Tabela 4.1: Editores XML e algumas de suas características.

epcEdit

A ferramenta ***epcEdit*** mostrou conter um número maior de qualidades reunidas em um só produto em relação aos demais editores analisados.

O editor epcEdit facilita a edição de grandes documentos XML ou SGML com grande controle da estrutura do documento através de uma interface visual do documento em questão (figura 4.5).

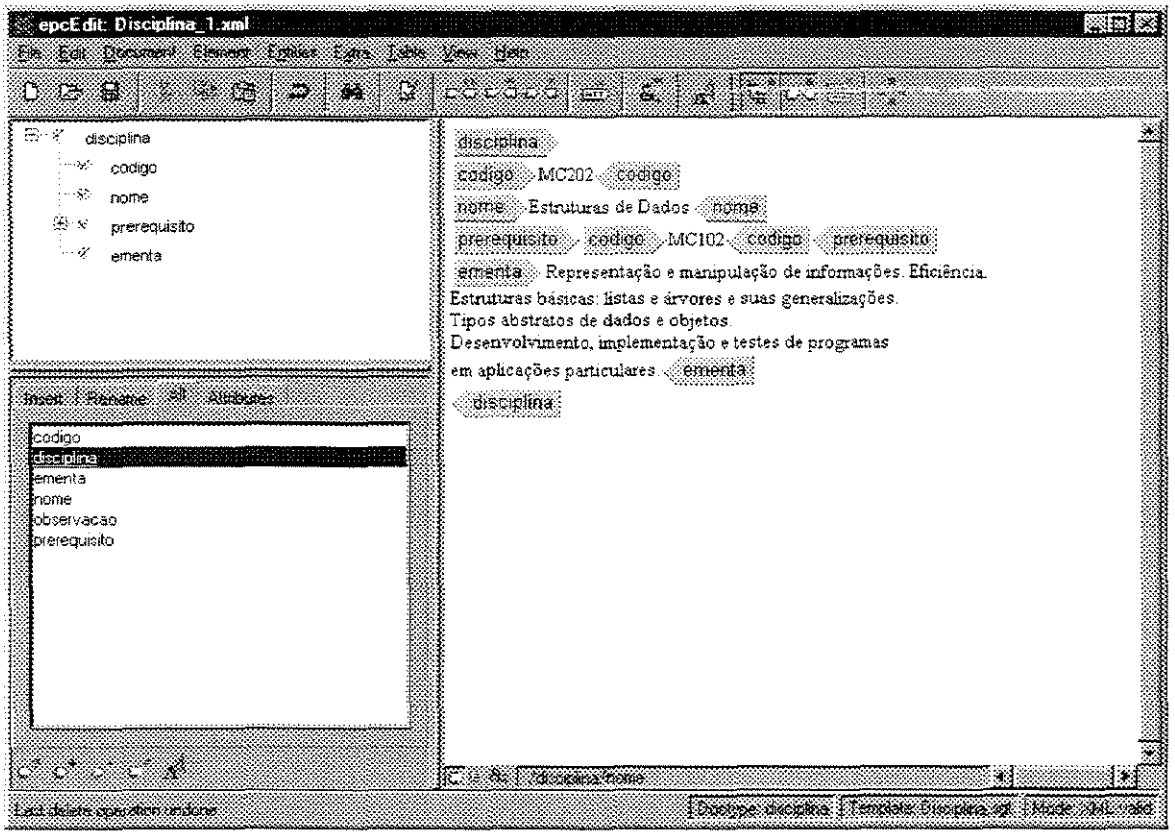

Figura 4.5: Visão geral do editor *epcEdit*.

A ferramenta contém um parser integrado para validação de documentos, seja para a verificação de um documento XML bem-formado ou para a validação a partir de um DTD, (figura 4.6) verificando possíveis erros mesmo durante a edição do documento.

A habilidade de executar uma validação completa (figura 4.7) simplifica a manutenção

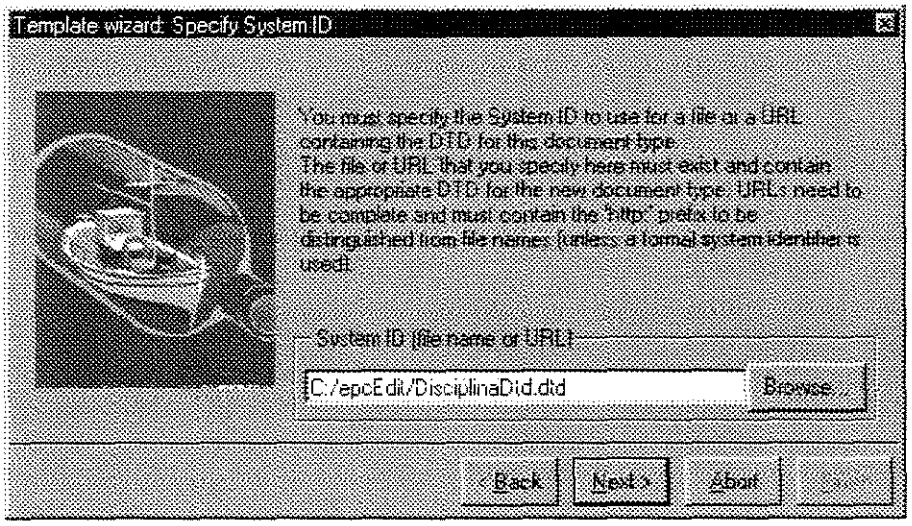

Figura 4.6: Escolha de um DTD para validação de documentos.

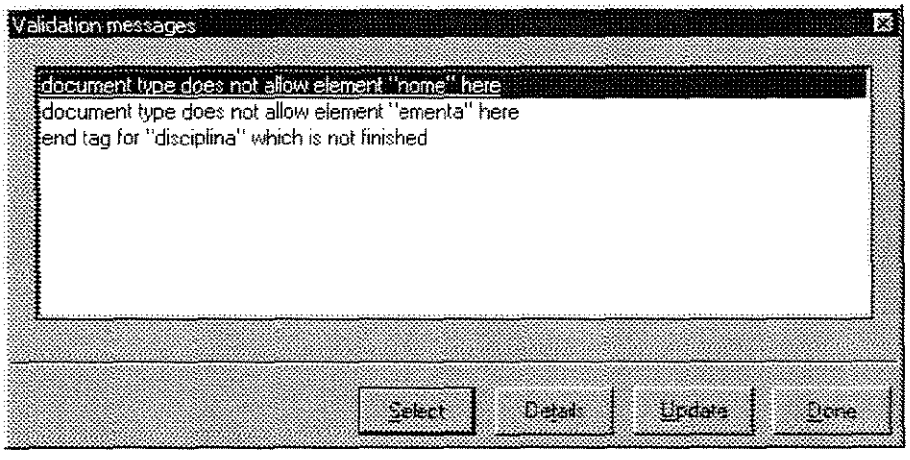

Figura 4.7: Mensagens de erros na validação do documento.

de documentos XML e permite a detecção de problemas potenciais, mesmo em documentos inválidos ou incompletos.

Possui um editor para tabelas HTML e um editor para inserir ou modificar os atributos contidos nos elementos, tornando mais fácil a manipulação desses elementos no documento XML.

O editor de atributos do epcEdit na figura 4.8 verifica se os valores dos atributos estão corretos, possuindo suporte especial para os atributos *ID* e *IDREF*, descritos anteriormente e utilizados para identificação de elementos e fazer referências a esses elementos evitando a re-escrita de suas informações.

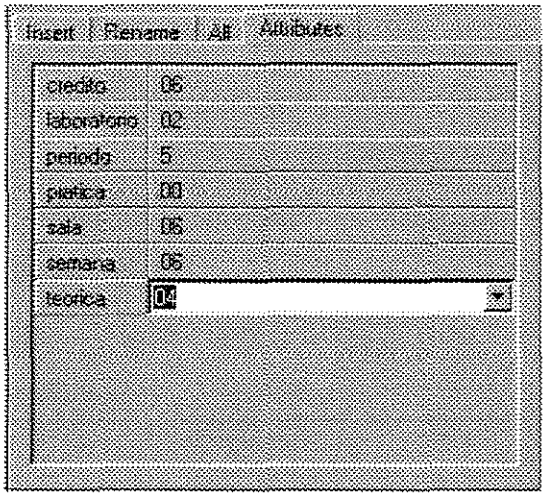

Figura 4.8: Edição dos atributos de um elemento.

A estrutura hierárquica de um documento é representada por uma visão em árvore que é atualizada durante as mudanças que ocorrem no documento.

O editor possui uma visualização através de diagramas que permitem ao usuário compreender a estrutura das construções no DTD utilizado. A ferramenta provê informações sobre o documento XML, o conteúdo permitido nos elementos do documento e seus atributos, contidos no modelo do DTD, forçando uma utilização consistente dos elementos e atributos mesmo em documentos sem DTD (bem-formados), mostrando ser uma ferramenta

com várias funcionalidades e de fácil utilização.

EditML Pro

É outro editor XML que também mostrou vários pontos positivos.

EditML Pro é um editor visual para a criação de arquivos XML, DTD e folhas de estilo CSS. Também fornece suporte para folhas de estilo XSL.

Possibilita a verificação de documentos XML bem-formados e documentos XML válidos a partir de um DTD.

Em sua interface gráfica (figura 4.9) é possível editar e alterar documentos utilizando visualização em árvore dos elementos, mostrando toda a hierarquia entre os elementos. Esse editor também fornece uma visualização da lista de atributos e do conteúdo de um elementos selecionado para edição do seu conteúdo.

Também é possível a visualização do documento para editar ou alterar seu conteúdo totalmente em modo texto. Para uma fácil identificação, as diferentes características do documento são mostradas em cores diferentes para os tags, atributos e conteúdo. Os tags finais de cada elemento são gerados automaticamente quando um tag de um elemento é criado.

É possível visualizar o DTD correspondente ao documento XML, como também a criar um novo DTD para arquivos XML bem-formados.

A visualização de um documento XML para a Web pode ser feita através da utilização de folhas de estilos CSS e XSL dentro do próprio editor EditML Pro, selecionando uma folha de estilo como mostra a figura 4.10. É possível a edição de folhas de estilos CSS a partir de uma nova janela em modo texto, como também editar arquivos XSL, já que esses documentos obedecem ao padrão XML.

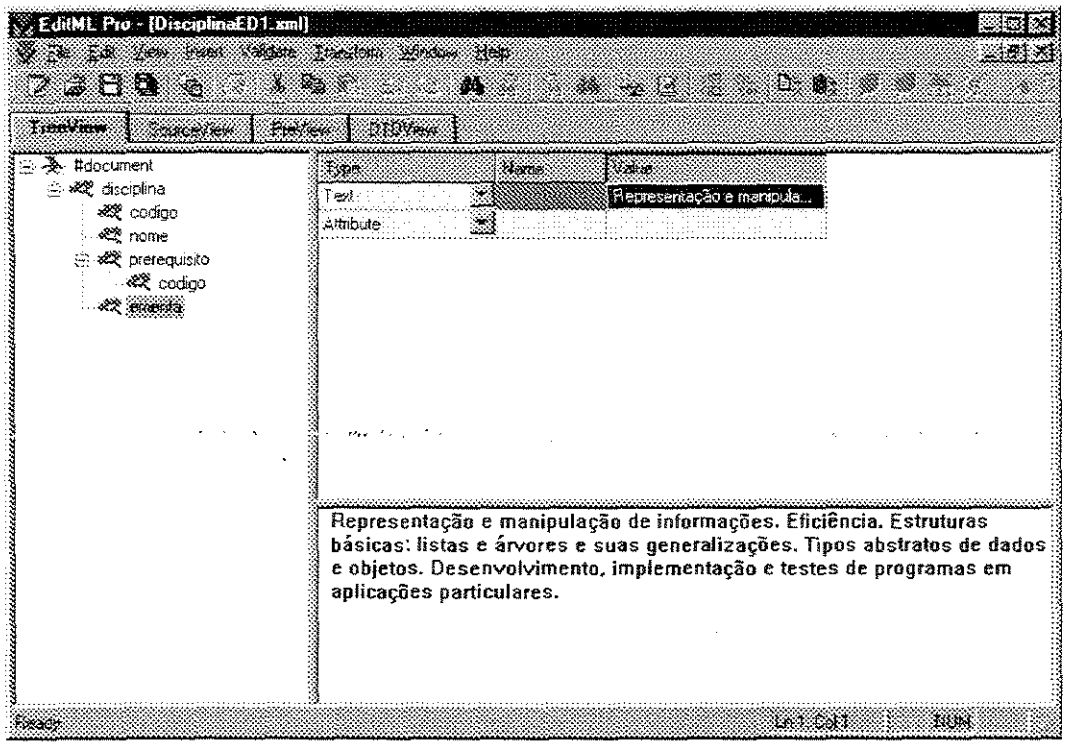

Figura 4.9: Visão geral do editor *EditML Pro*.

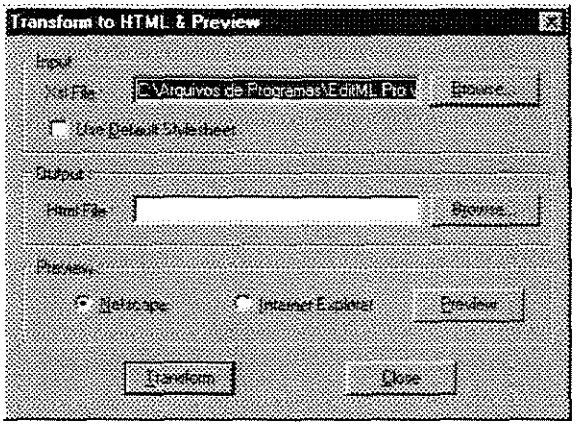

Figura 4.10: Visualização do código XML.

Athens XML Editor

O Athens XML Editor está disponível na plataforma Windows como um editor XML *shareware*.

A ferramenta Athens XML Editor possibilita a visualização da hierarquia entre os elementos presentes no documento e o conteúdo inicial de cada elemento. A edição da informação contida em um documento XML é realizada através de uma interface praticamente em modo texto, ilustrada na figura 4.11.

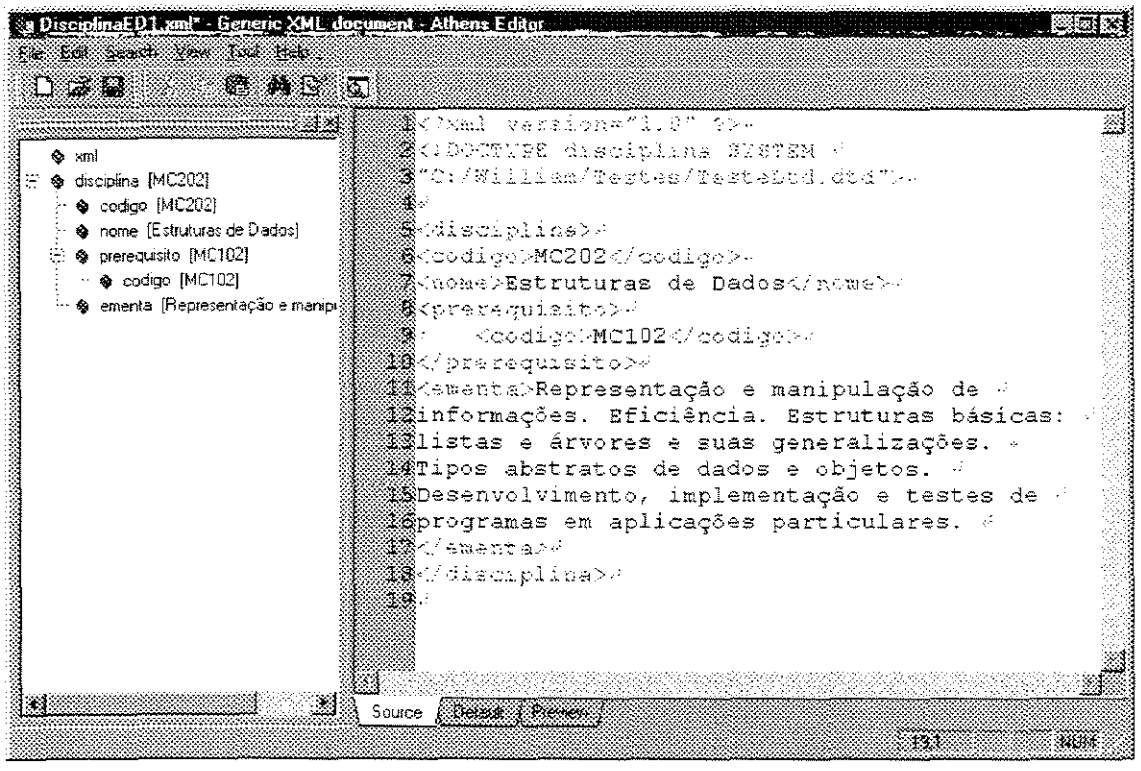

Figura 4.11: Visão geral do editor *Athens XML Editor*.

Entre as características desse editor citamos a conclusão automática dos nomes de tags e atributos e dos tags finais de cada elemento, verificação dinâmica se o documento XML está bem-formado, coloração diferenciada para cada característica do documento (elementos e atributos) e também a transformação de documentos XML em arquivos HTML utilizando

folhas de estilos XSL.

É possível fazer uma análise da correção do documento XML através de seu respectivo DTD, como mostra a figura 4.12.

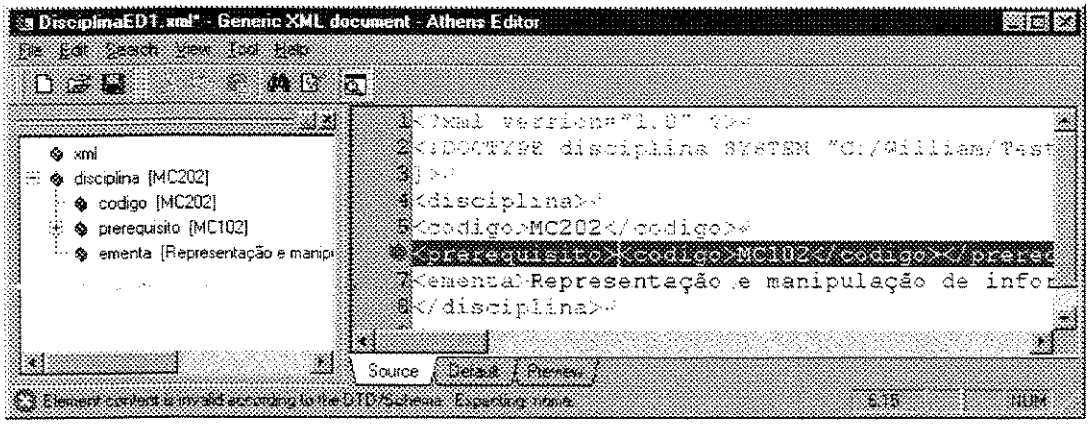

Figura 4.12: Análise do documento XML de acordo com seu DTD.

Também é possível utilizar esse editor para a criação ou alteração das regras definidas em um DTD. O modo textual de edição do DTD (figura 4.13) também segue as características disponíveis para a criação de documentos XML.

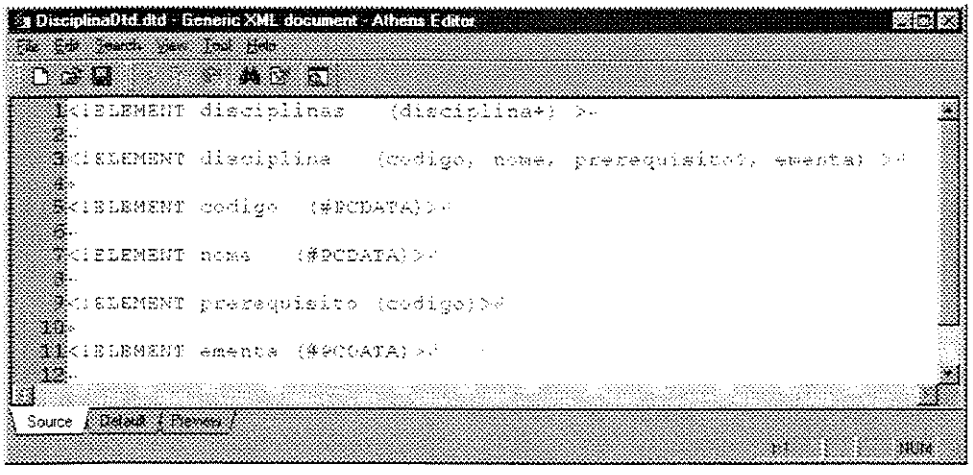

Figura 4.13: Edição de um DTD com o Athens XML Editor.

As funcionalidades desse editor já são mais restritas comparadas aos editores vistos anteriormente, mas ainda é razoável.

Peter's XML Editor

O editor ***Peter's XML Editor*** permite a edição de documentos com o auxílio de uma visão em árvore da hierarquia de elementos no documento XML, também a edição de seu conteúdo através do modo texto do arquivo, utilizando cores variadas para uma melhor identificação de elementos e atributos.

Fornece habilidade para adicionar, editar e apagar elementos, atributos, texto, seções CDATA, comentários e instruções de processamento através da estrutura hierárquica dos elementos e seus respectivos conteúdos como mostra a figura 4.14.

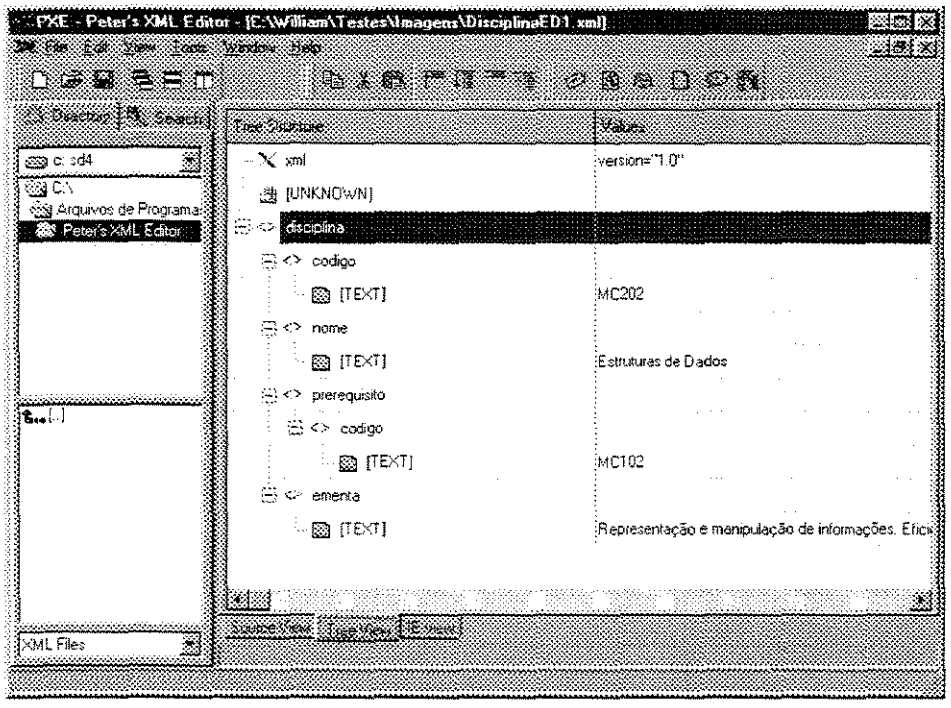

Figura 4.14: Visão geral do editor *Peter's XML Editor.*

Também disponibiliza visualização e manipulação em modo texto do código contido no

documento XML, mostrado na figura 4.15, diferenciando elementos e atributos através de cores distintas e utilizando uma conclusão automática dos tags finais dos elementos.

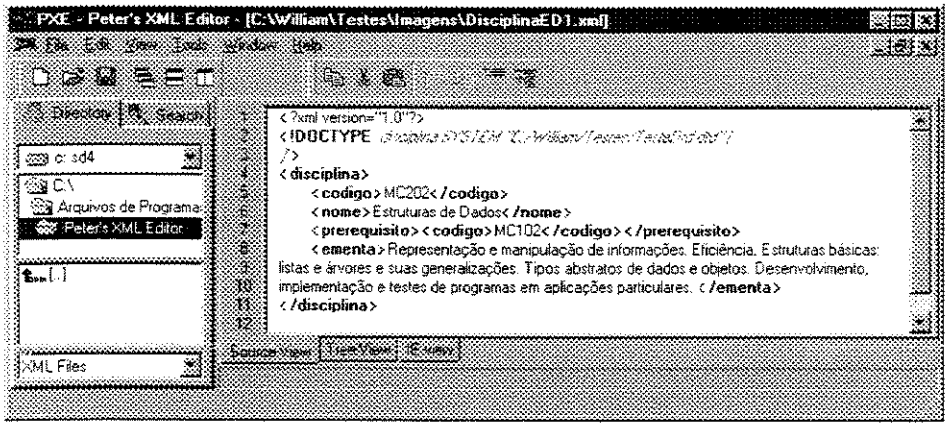

Figura 4.15: Visualização em modo texto do documento XML.

É possível a visualização de um documento XML utilizando recursos do Internet Explorer, como mostra a figura 4.16

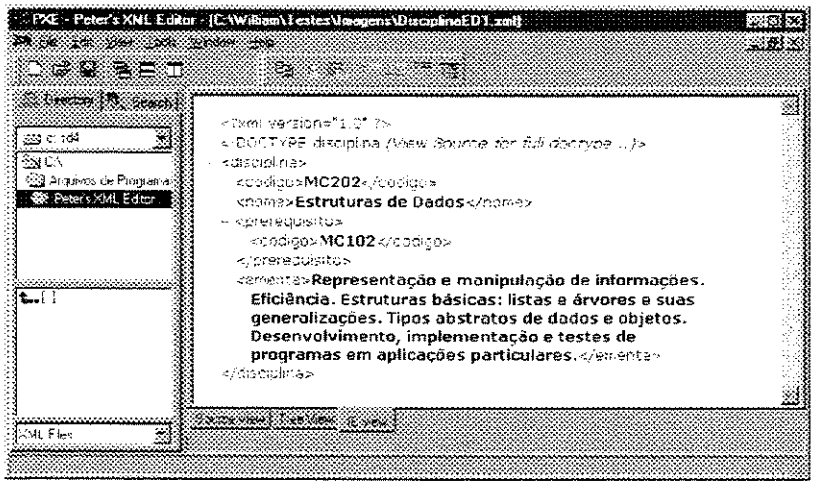

Figura 4.16: Visualização do documento XML utilizando o Internet Explorer.

Esse editor utiliza uma interface visual sem maior facilidade de manipulação com relação aos editores anteriores, não possuindo alguns recursos como a edição de outros tipos de

documentos.

Cooktop

Cooktop é um editor para documentos XML, DTD, e de folhas de estilos XSL. Foi desenvolvido para a plataforma Windows e com licença *freeware*.

A edição de documentos XML, DTD e de folhas de estilos XSLT, é realizada em modo texto como mostra a figura 4.17, usando cores diferentes para separar as características do documento em edição.

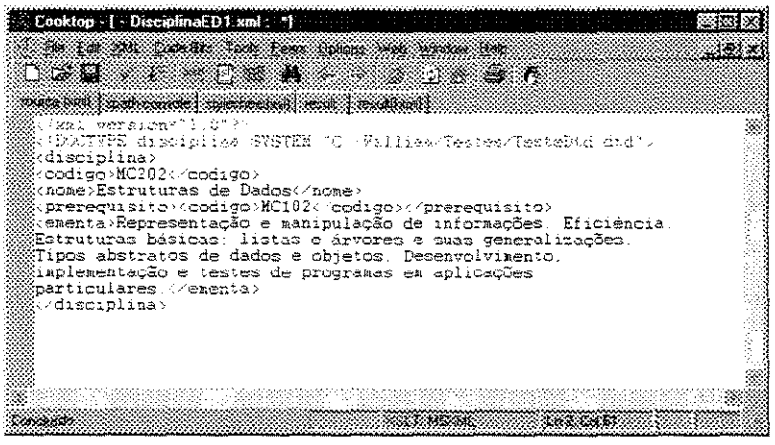

Figura 4.17: Visualização em modo texto do editor XML *Cooktop*.

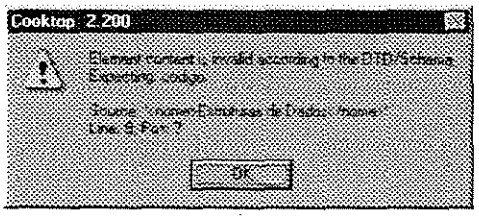

Figura 4.18: Mensagem de erro na validação do documento XML.

Também é verificada a correção do documento XML a partir de seu respectivo DTD e se o documento está bem-formado. Caso seja encontrado erro na validação do documento é

exibida uma mensagem ao usuário como mostra a figura 4.18.

Além da edição de folhas de estilos XSLT é possível visualizar o resultado final do documento utilizando um processador XSLT para gerar documentos HTML, XHTML ou outro documentos XML.

O editor não possui características visuais de edição de documentos, dificultando a criação e a modificação de seus conteúdos.

4.2.2 Utilização de Formulários Web

Outra alternativa analisada foi a utilização de formulários Web desenvolvidos especificamente para uma aplicação do modelo.

Uma das vantagens de se utilizar formulários Web sobre editores XML é que os usuários que fornecerão as informações não terão que se preocupar com as formatações definidas no XML pelo seu DTD: por exemplo, quais são os elementos (tags) existentes, qual a estrutura hierárquica entre esses elementos, quais elementos são referentes a uma certa informação que deve ser fornecida, etc, além de não terem que aprender a utilizar um editor XML; basta preencher os campos do formulário com as informações solicitadas. Assim os usuários do sistema não precisam ter conhecimento da linguagem XML e de seu DTD.

O desenvolvedor dos formulários pode ainda verificar possíveis erros ou restrições na edição das informações para a criação do documento XML: por exemplo, a falta de alguma informação obrigatória ou erro de digitação das informações, podendo enviar mensagens ao usuário sobre o erro ocorrido, informando como deve ser feita a entrada de uma certa informação.

A seguir será dada uma introdução sobre o processamento de formulários Web.

Processamento de formulários em Servidores Web

Para entendermos como funciona o processamento de formulários em servidores Web é necessários saber o que é CGI (Common Gateway Interface) [20]: através de programas ou

scripts CGI é possível fazer o processamento de formulários Web, o acesso a banco de dados, e gerar páginas Web dinâmicas além de muitas outras facilidade.

CGI é uma interface genérica para executar programas externos (gateways) suportados por um servidor de informação, especificando a forma como os dados devem ser transferidos entre um servidor Web e programa externo ao servidor e vice-versa, incluindo variáveis de ambiente como o nome do gerador da informação, entre outras.

Os servidores HTTP são os servidores de informação que dão suporte aos scripts CGI e utilizam um protocolo específico para o envio de documentos para os clientes, que são os Browsers. CGI é o mecanismo geral com o qual o servidor lida com informações recebidas de clientes.

Existem três componentes básicos para a manipulação das informações remetidas pelo cliente:

- Um programa na máquina do servidor para processar a informação.
- Um mecanismo através do qual o servidor HTTP possa passar as informações recebidas para esse programa.
- Um meio desse programa retornar as informações para o cliente (browser) via servidor http para que o usuário veja o resultado.

A figura 4.19 ilustra o funcionamento de scripts (podendo ser tanto um programa compilado como um script interpretado) que tratam as requisições vindas dos clientes (browsers). As requisições são processadas e são gerados documentos HTML que são enviados de volta para os clientes. Os clientes interpretam os documentos e os apresentam na tela. Observe que não necessariamente o script reside numa máquina distinta do servidor Web.

Um software que segue esse padrão não está limitado a uma linguagem específica, pois o padrão é suficientemente genérico para permitir implementações nas mais diversas combinações de ambientes e linguagens. Há muitas bibliotecas disponíveis para C, Perl e PHP.

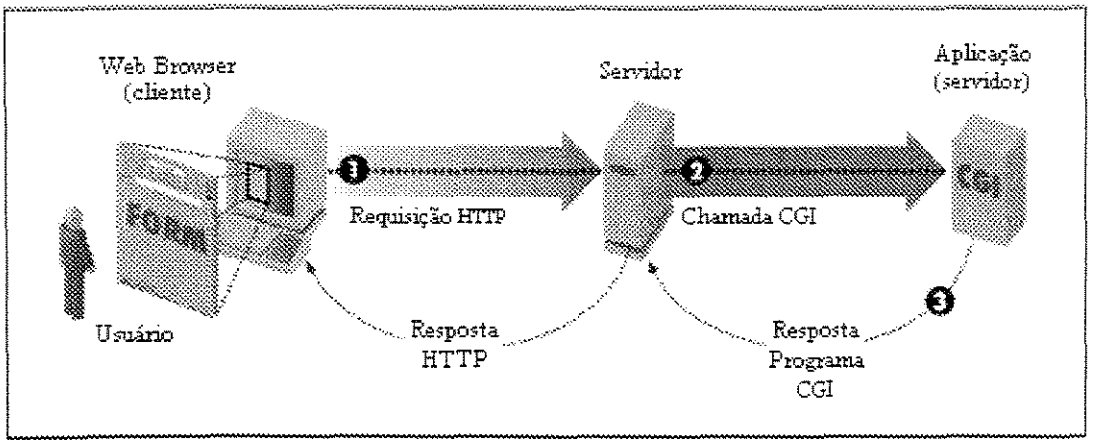

Figura 4.19: Formulário para a geração de documento XML.

CGI permite dar dinamismo às páginas Web aumentando o nível de interação com os usuários, sendo bastante utilizado para as seguintes aplicações:

- Processamento de formulários enviados pelos browsers, possibilitando consultas a bancos de dados, cadastro de clientes, visitantes, pesquisas de opinião, etc.
- Contadores de acesso a páginas e relógios.
- Exibição de textos e imagens aleatórias, geração de documentos personalizados em tempo de execução.

Os servidores Web possuem a capacidade de receber dados dos clientes através dos métodos GET e POST, utilizados para submeter formulários ao servidor.

Normalmente, GET é usado para obter um arquivo ou outro recurso: é como o browser faz download da maioria de arquivos como HTML e imagens. Ele também pode ser usado para as submissões de formulários, se não houver dados demais (embora seja recomendável, por razões de privacidade dos dados enviados).

Quando um formulário HTML é submetido usando POST, os dados do formulário são anexados ao final da solicitação POST em seu próprio objeto. Por exemplo, é possível enviar

arquivos inteiros usando POST, pois o tamanho dos dados não é limitado, como ocorre com GET.

Existem duas formas básicas de passar os dados para os scripts CGI:

- Variáveis de ambiente
- Entrada padrão

No contexto das *variáveis de ambiente* existem diversas variáveis padronizadas que sempre estão disponíveis para o script CGI. Entre elas estão disponíveis variáveis de ambiente com usos interessantes como o tipo de conteúdo, protocolo utilizado, nome e endereço do host do cliente, browser que ele está usando, método usado para chamada do programa, nome do servidor e etc.

A outra forma de passagem de dados para os scripts (*entrada padrão*) serve para submissão de requisições de formulários (FORM) via método POST. Os dados chegam em uma cadeia na forma de pares *nome=valor*, separados pelo símbolo &. Cada par está codificado ("URL encoded"), onde os espaços estão substituídos por "+" e alguns caracteres codificados em hexadecimal precedidos por um sinal de "%". Os pares *nome=valor* são formados pelo browser quando o usuário submete um formulário, usando os nomes dos campos dos formulários e os valores associados a estes campos pelo usuário para criar os pares.

A seguir será feito um comentário sobre as linguagens de programação Perl, Python e PHP utilizadas na programação de aplicativos Web.

Perl

A linguagem Perl [33] permite a criação de programas em ambientes UNIX, MSDOS, Windows, Macintosh, OS/2, e em outros sistemas operacionais.

Trata-se de uma linguagem que possui funções eficientes para manipulação de textos, o que a torna muito popular para programação de formulários Web, além de ser muito utilizada em tarefas administrativas de sistemas UNIX, em lugar das linguagens *"shell"*.

A linguagem Perl é de fácil utilização na programação de uma ampla faixa de tarefas,

particularmente aquelas que envolvem manipulação de arquivos de texto. A figura 4.20 ilustra um simples exemplo de uso da linguagem Perl.

```
#!/usr/bin/perl

print "Content-type:text/html\n\n";

print "<html><head><title>Test Page</title></head>\n";
print "<body>\n";
print "<h2>Hello, world!</h2>\n";
print "</body></html>\n";
```

Figura 4.20: Exemplo de um simples script Perl.

Em síntese, a utilização dessa linguagem na implementação de projetos se deve principalmente por sua extrema capacidade em manipular arquivos de texto e pela boa interação que possui com a Internet, recursos que se sobressaem às possíveis desvantagens citadas.

Python

Python [31] é uma linguagem de alto nível interpretada, orientada a objetos com uma semântica dinâmica. Suas estruturas de alto nível, combinadas com sua tipagem de amarração dinâmica a torna muito atrativa para desenvolvimento de aplicativos para uso como linguagem de script.

Python pode ser estendido de uma forma sistemática adicionando novos módulos implementados em uma linguagem compilada como C ou C++. Essas extensões podem definir novas funções e variáveis como também novos tipos de objetos.

A sintaxe simples do Python encoraja a reutilização de código simplificando a manutenção e a normalização de dados em módulos e pacotes distintos. Esta linguagem e seu código fonte encontra-se gratuitamente disponível na Internet.

PHP

PHP (Hypertext Preprocessor) [3] é uma linguagem de programação interpretada voltada para o desenvolvimento de aplicações para a Web.

O objetivo principal da linguagem é permitir que desenvolvedores criem páginas HTML geradas dinamicamente, a partir de um servidor Web.

No exemplo da figura 4.21 nota-se como PHP é diferente de scripts CGI escritos em Perl ou C: ao invés de escrever um programa com vários comandos para imprimir códigos HTML, em PHP é possível escrever um arquivo HTML com algum código inserido para fazer alguma coisa (nesse caso simples, imprimir um pouco de texto). O código PHP é delimitado pelo tag inicial ("<?php") e final ("?>") que permitem escrever código dentro e fora do modo de interpretação do PHP.

```
<html>
<head>
  <title>Exemplo</title>
</head>
<body>
  <?php
    echo "Um teste de Script PHP!";
  ?>
</body>
</html>
```

Figura 4.21: Exemplo introdutório de PHP.

A diferença de PHP com relação a linguagens semelhantes a Javascript é que o código PHP é executado no servidor Web, não no cliente como ocorre com Javascript, sendo enviado para o cliente apenas HTML puro. Se um script similar ao anterior for executado em um servidor, o cliente receberia os resultados da execução desse script sem nenhum modo de determinar qual é o código fonte que o gerou.

Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor,

com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

Por ser uma linguagem script executada no servidor, PHP pode fazer qualquer coisa que um programa CGI faz: coletar dados de formulários, gerar conteúdo dinâmico de páginas ou enviar e receber cookies.

PHP, assim como Perl, tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como Interbase, mySQL, Oracle, Sybase e vários outros.

Além disso, PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir sockets e interagir com outros protocolos.

PHP é bastante poderoso em recursos de processamento de texto, e como interpretador e estruturador para documentos XML, possuindo bibliotecas específicas para tais manipulações. Para acessar e processar documentos XML, são suportados os padrões DOM (Document Object Model) [14] e SAX (Simple Api for Xml) [25], que são interfaces da linguagem PHP que auxiliam na manipulação de documentos XML.

Utilização do padrão DOM sobre a linguagem PHP

Para facilitar a entrada de dados fornecida por usuários distribuídos em uma intranet foram criados formulários Web utilizando a linguagem PHP, como por exemplo o formulário da figura 4.22. A linguagem PHP disponibiliza bibliotecas para o uso de funções relacionadas ao padrão DOM, utilizado na manipulação de dados estruturados em XML, como a criação e alteração da informação em documentos XML.

A extensão DOM do PHP possibilita a manipulação de documentos XML. Por exemplo, é possível através da função *domxml_xmltree()* construir o documento XML completo em uma estrutura de árvore. Um documento XML pode ser lido e modificado utilizando funções como *DomDocument_create_element()* – utilizada para a criação de um novo elemento, *DomDocument_create_text_node()* – usado para a criação de elementos que contenham

texto, *set_attribute()* - utilizado para alterar o valor de algum atributo de certo elemento, etc.

As informações adquiridas através do formulário da figura 4.22 são enviadas através do método POST para a parte do formulário que irá processar esses dados, através do comando:

<form action="ProcessaXML.php" method="post" enctype="multipart/form-data">.

As informações fornecidas nos formulários pelos usuários serão reconhecidas e manipuladas através dos nomes dados aos respectivos campos definidos no formulário.

O arquivo responsável pelo processamento dos dados fornecidos e pela criação do documento XML dentro do padrão necessário pela aplicação do modelo, chamado *"ProcessaXML.php"*, utiliza-se de funções do padrão DOM que serão mostrados a seguir com a intenção de exemplificar como foi realizada a criação de um documento XML utilizando funções DOM com a linguagem PHP.

Inicialmente é feita a abertura de um arquivo XML vazio, no caso da criação de um novo documento XML, ou de um arquivo XML que já contenha informações estruturadas e que serão alteradas pelo formulário Web:

```
<?php
$fp = fopen ("Arquivo.xml", "w"); // Abre o arquivo XML para escrita.
$doc = new_xmldoc("1.0"); // Cria a instrução de processamento XML.
$root = $doc->add_root("disciplina"); // Adiciona um elemento raiz "disciplina".
?>
```

Em seguida é criado o cabeçalho do arquivo que identificará um documento XML com a instrução de processamento através do comando *"new_xmldoc"*, passando como parâmetro a versão do XML.

Para adicionarmos o elemento raiz ao documento XML é utilizado o comando *"add_root"*, onde é passado como parâmetro o elemento raiz do documento XML.

No processamento do código da disciplina é feita uma verificação de sua validade em relação às regras utilizadas pela Unicamp através da função *"CodigoCorreto()"*. Se o código da disciplina estiver correto um novo elemento é criado e adicionado internamente ao elemento

raiz do documento XML através do comando *"new_child"*, onde é passado como parâmetro o nome e o conteúdo desse novo elemento; caso contrário uma mensagem de erro é enviada ao usuário:

```
<?php
if (CodigoCorreto($codigo))
    echo "<b>$codigo</b> ";
    $children = $disciplina->new_child("codigo", $codigo); // Cria um elemento filho.
else echo "<p><center><font size=3 color=red>O <b>código</b> da disciplina está
incorreto!</font></center>";
?>
```

Para o nome da disciplina é testado apenas se ele foi informado pelo usuário, sendo criado e adicionado mais um elemento internamente ao elemento raiz do documento XML:

```
<?php
if ($nome <> "")
     echo "<b>$nome</b>";
    $disciplina->new_child("nome", $nome); // Cria um novo elemento filho.
else echo "<p><center><font size=3 color=red>O <b>nome</b> da disciplina
está em branco!</font></center>";
?>
```

O pré-requisito da disciplina é um elemento opcional. Primeiramente é testado se o pré-requisito foi informado e então é criado e adicionado um elemento ao documento XML. Posteriormente serão separados e analisados os códigos das disciplinas de pré-requisito: para cada código válido um novo elemento é criado e adicionado ao elemento pré-requisito. Em caso de erro uma mensagem é enviada ao usuário:

```
<?php
$CodigoErrado=0;
if (strlen($prerequisito) != 0)
    $i=0;
    $children = $disciplina->new_child("prerequisito", ""); // Cria um elemento filho.
    while ($i < strlen($prerequisito))
      if (($prerequisito[$i] != " ") and ($prerequisito[$i] != "/") and
        (strlen(substr($prerequisito,$i)) >= 5))
        $prerequisito[$i] = strtoupper($prerequisito[$i]);
        $prerequisito[$i+1] = strtoupper($prerequisito[$i+1]);
        $ChildrenCodigo = $children->new_child("codigo", substr($prerequisito, $i, 5));
        // Cria um novo elemento filho.
        $i=$i+5;
        if (!CodigoCorreto($parte)) $CodigoErrado=1;
      else if ($prerequisito[$i] == "/")
          $children->set_content("/");
        elseif ($prerequisito[$i] == " ")
          $children->set_content(" ");
        elseif (strlen(substr($prerequisito,$i)) < 5)
          $CodigoErrado=1;
        $i=$i+1;
    if ($CodigoErrado == 1)
        echo "<p><center><font size=3 color=red>O <b>código</b> de alguma
        disciplina de <b>pré-requisito</b> está incorreto!</font></center>";
    else echo "<b>Pré-requisito : </b><font color='#000000'>";
        $prerequisito
        echo "</font>";
?>
```

A ementa da disciplina é testada caso tenha sido informada, sendo criado um novo elemento, adicionado ao elemento raiz do documento XML:

```
<?php
if ($ementa <> "") echo "<b>Ementa : </b><font color=#000000>$ementa</font>";
    $disciplina->new_child("ementa", $ementa); // Cria um novo elemento filho.
else echo "<p><center><font size=3 color=red>A <b>ementa</b> da disciplina está em branco!</font></center>";
?>
```

Finalmente, a estrutura XML criada é passada para a variável *"xmlfinal"* para ser escrita no arquivo XML, completando o processamento da informação enviada pelo formulário Web:

```
<?php
$xmlfinal = $doc->dumpmem(); // Passa toda a estrutura que foi criada na
memória para a variável $xmlfinal.
fwrite ($fp, $xmlfinal); // Escreve no arquivo "Arquivo.xml" toda estrutura XML
contida na variável $xmlfinal.
fclose($fp); // Fecha o arquivo "Arquivo.xml".
?>
```

Ambientes de programação para a Web

Os sistemas de integração à Web estão se especializando de tal forma que cada vez mais os programas se distanciam da interface simples do CGI.

É o caso do servidor Web IIS (Internet Information Services) da Microsoft para a plataforma Windows, que suporta a tecnologia ASP (Active Server Pages) combinando CGI e pré-processamento de páginas. Também para o ambiente Windows e Linux temos o Cold Fusion, combinando código de programação e pesquisas SQL em um único bloco. Ambos são proprietários.

O servidor Web Apache é um ambiente livre, disponível para as plataformas Windows e Linux, integrando linguagens como PHP, Perl e Python, funcionando como módulos do

Apache. PHP é sintaticamente parecido com a linguagem C mas com maior flexibilidade principalmente no nível de manipulação de strings, além de conter uma extensa biblioteca para os mais diversos fins, como as bibliotecas utilizadas na manipulação da informação em XML, utilizadas nesse trabalho.

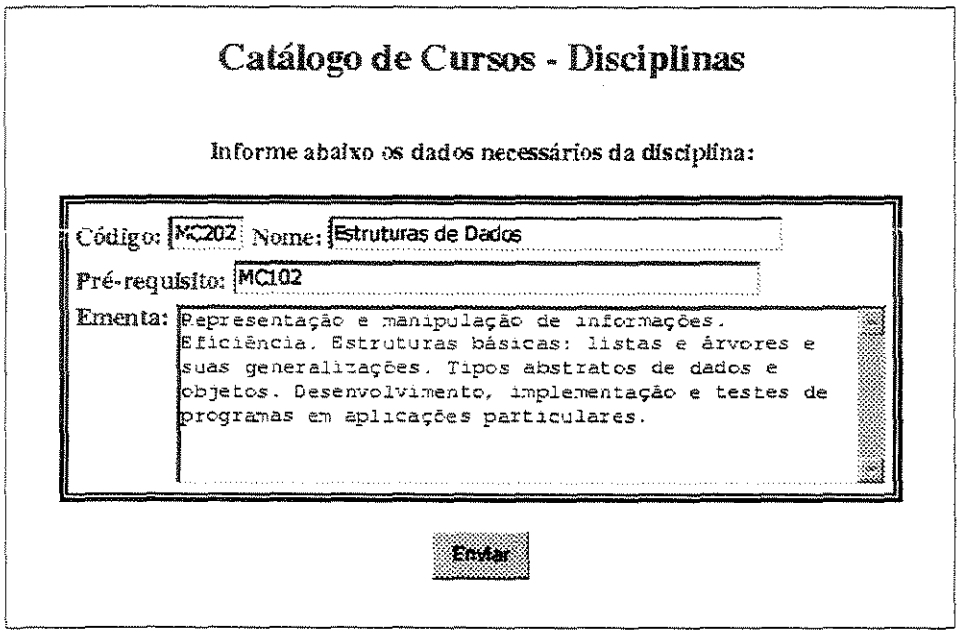

Figura 4.22: Formulário para a geração de documento XML.

No formulário da figura 4.22 o usuário preenche os dados de uma disciplina. A disciplina possui dados que serão solicitados pelo formulário como o código da disciplina, seu nome, o código das disciplinas pré-requisito, e uma descrição da ementa informando o que será abordado durante o curso da disciplina.

Desta forma, usuários do sistema não precisam ter conhecimento sobre a linguagem XML.

As informações fornecidas serão submetidas pelo usuário através do botão *Enviar* para um servidor Web, responsável por processar os dados descritos no formulário, criando ou alterando parte de um documento XML.

4.3 Conversão das Informações XML

Nesta parte do processo, o documento padrão XML será processado pela ferramenta "Jade" [11] (James' DSSSL Engine), que é uma ferramenta que utiliza a linguagem de estilo DSSSL (Document Style Semantics and Specification Language) [22] e XSL (Extensible Style Language) [2], especificando como será a visualização do documento final, tanto de um documento para a Web (HTML) quanto para documentos do tipo texto (RTF, PS e PDF), para impressão.

O formato HTML e XHTML utiliza uma folha de estilo DSSSL desenvolvida especificamente para esse formato, diferente de uma folha de estilo DSSSL para um arquivo de impressão (formatos PS e PDF).

Para exemplificar o processo de conversão utilizaremos como exemplo a estruturação em XML dos dados de uma disciplina, mostrados na figura 4.1. Utilizaremos a folha de estilo DSSSL exibida na figura 4.25 para representar a formatação da disciplina no formato PS.

A ferramenta Jade necessita de dois documentos de entrada: um documento XML, e um documento definindo uma folha de estilo DSSSL apropriada para essa aplicação. Como resultado final foi gerado o documento para impressão no formato PS visualizado na figura 4.23. Também é possível gerar documentos no formato PDF, e no formato HTML, onde a figura 4.24 representa a sua visualização através de um browser.

Jade é uma ferramenta "Open Source" disponível para as plataformas Windows e Linux. Ao ser executada pela linha de comando, "jade", possui entre outras opções:

- -d *arquivo-dsssl* : especifica qual arquivo DSSSL contém as definições de visualização;
- -t *tipo-arquivo* : especifica o tipo de saída desejado, por exemplo, "rtf" ou "tex".

Exemplo de execução da ferramenta Jade:

- *jade -t tex -d folha-estilo.dsl documento.xml > documento.tex*

Observe que na verdade Jade produz uma saída em *tex*. Os formatos PS e PDF são gerados por pacotes adicionais disponíveis no ambiente Latex.

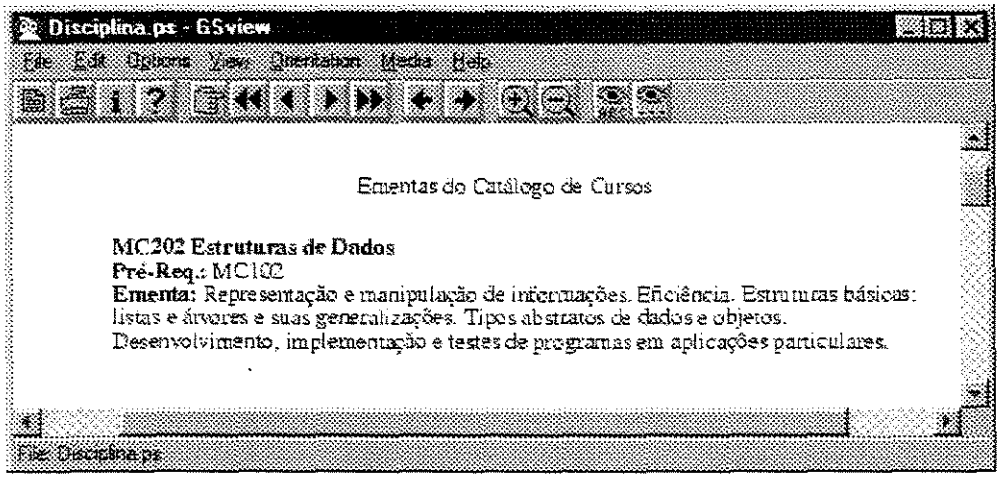

Figura 4.23: Exemplo de visualização de documento PS utilizando uma folha de estilo DSSSL.

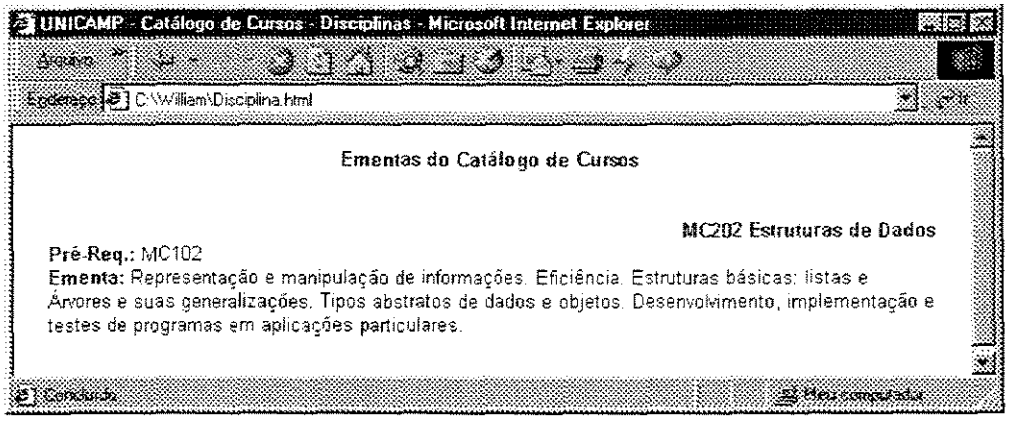

Figura 4.24: Exemplo de visualização de documento HTML utilizando uma folha de estilo DSSSL.

4.3.1 Jade - James' DSSSL Engine

Jade (James' DSSSL Engine) [11] é uma ferramenta de processamento da linguagem de estilo DSSSL. Foi desenvolvida para as plataformas Unix e Windows.

A ferramenta Jade inclui o seguintes componentes funcionais:

- Uma interface abstrata
- Uma implementação em memória de sua interface. Possui um processamento multi-threaded, permitindo que parte dos resultados possam ser acessados antes de sua execução completa e o acesso as propriedades de cada nós estarão bloqueados até que essa propriedade se torne disponível, possibilitando ao processador de estilo produzir uma saída antes da leitura de todo o documento.
- Um processador de estilo que implementa a linguagem de estilo DSSSL. O processador de estilo utiliza o formato de saída usado para a formatação do documento. Mas não fornece uma interface que permita a edição de documentos XML ou folhas de estilos DSSSL.
- Uma aplicação manipulada pela linha de comando *jade* que combina o processador de estilo com as seguintes possibilidades de tipos de documentos finais:
  - A geração de documentos XML utilizando diferentes elementos e alterando a hierarquia existente entre os elementos.
  - A geração de documentos no formato RTF, utilizado pela plataforma Windows.
  - A geração de documentos TeX, sendo posteriormente processado podendo gerar documentos no formato PS e PDF.
  - A geração de documentos SGML, permitindo sua utilização em transformações SGML.

Na linha de comando da ferramenta Jade é possível especificar algumas opções como:

- **-d** – Especifica qual a folha de estilo DSSSL será utilizada no processamento.
- **-t** – Indica qual será o tipo do formato de saída desejado utilizando a folha de estilo apresentada. Podendo ser: *rtf*, *tex*, *sgml* e *xml*.
- **-o** – Fornece qual será o arquivo que receberá o resultado de saída do processamento. Se esse atributo não for utilizado será criado um arquivo de saída que receberá o nome documento de entrado XML.

Se o documento de entrada XML possuir um DTD especificado em seu conteúdo, a ferramenta Jade executará uma validação do mesmo advertindo o usuário sobre eventuais erros presentes nos arquivos utilizados, como também no arquivo DSSSL, que torne impossível o processo de conversão do documento XML.

4.3.2 Especificação DSSSL de uma Ementa do Catálogo de Cursos

DSSSL (Document Style Semantics and Specification Language) [22] é uma linguagem de estilo, para definir os estilos que serão aplicados ao conteúdo de um documento XML. É possível determinar a posição de um certo conteúdo na organização física das páginas de um documento (por exemplo: centralizar o texto), selecionar e reordenar o conteúdo que irá aparecer no documento final, alterar o tipo e o tamanho da fonte utilizada, incluir partes que não estavam presentes no documento original (por exemplo: cabeçalho e rodapé), ou ocultar no documento final informações presentes no documento original.

A figura 4.25 é um exemplo de uma folha de estilo DSSSL definida para a disciplina do nosso exemplo. Utilizando a ferramenta Jade será obtido o documento para a visualização dessas informações no formato PS representado pela figura 4.23.

Na figura 4.25 a apresentação do elemento *disciplina* no documento final, inicia com um parágrafo que utilizará a fonte do tipo "Arial", o seu conteúdo em negrito ("bold"), o texto estará alinhado à esquerda e o tamanho da fonte utilizada será de 12pt. A instrução *(process-children)* também aplicará essas definições aos elementos filhos de *disciplina* (*codigo*, *nome* e *pre-requisito ementa*), a menos que um de seus elementos filhos modifique uma destas

```
(element disciplina
      (make paragraph        // Criação de um parágrafo.
            font-family-name: "Arial"    // Definição do tipo de fonte utilizada.
            font-weight: 'bold     // Especificação para o uso da fonte em negrito.
            quadding: 'start       // Texto alinhado à esquerda.
            font-size: 12pt        // Tamanho da fonte utilizada.
            (process-children)))     // Fazer o processamento dos demais elementos
                        // "filhos" da disciplina (ex.: nome e ementa da disciplina).
(element codigo
      (process-children))

(element nome
      (process-children))

(element pre-requisito
      (make paragraph
            (make sequence
                  (literal "Pré-Req.: "))
            (make sequence
                  font-weight: 'medium
                  (process-children))))

(element ementa
      (make paragraph
            (make sequence
                  (literal "Ementa: "))
            (make sequence
                  font-weight: 'medium)))
```

Figura 4.25: Exemplo de uma folha de estilo DSSSL para uma disciplina no formato PS.

definições para apresentar o seu próprio conteúdo, como acontece nos elementos *pre-requisito* e *ementa* que retira o uso do texto em negrito com a definição *font-weight: 'medium*, além da inserção de um texto fixo na saída do documento usando o comando *(literal "Ementa:")*.

A figura 4.26 é um exemplo de uma folha de estilo DSSSL para o formato HTML, definindo a criação dos elementos *
*, * * e *<font>*, para a disciplina do nosso exemplo. Utilizando Jade será obtido o documento HTML representado pela figura 4.24.

```
(element disciplina
   (make sequence
      (make element gi: "b"
         (make element gi: "div" attributes: '(("align" "right"))
            (make element gi: "font" attributes: '(("color" "#000099"))
               (make sequence
                  (process-matching-children "codigo")
                  (process-matching-children "nome")))))
      (process-matching-children "prerequisito")
      (process-matching-children "ementa")))

(element codigo (process-children))

(element nome (process-children))

(element pre-requisito
   (make element gi: "div"
      (make element gi: "b"
         (make element gi: "font" attributes: '(("color" "#000099"))
            (literal " Pré-Req.: ")))
      (process-children)))

(element ementa
   (make element gi: "div"
      (make element gi: "b"
         (make element gi: "font" attributes: '(("color" "#000099"))
            (literal " Ementa: ")))
      (process-children)))
```

Figura 4.26: Exemplo de uma folha de estilo DSSSL para uma disciplina no formato HTML.

Na figura 4.26 o elemento *disciplina* será apresentado no documento HTML final em negrito pela criação do elemento <*b*> ("bold"); será posicionado à direita pelo elemento <*div*> e a cor de sua fonte será alterada pelo elemento <*font*>; essas alterações serão aplicadas somente aos elementos *codigo* e *nome* do documento XML. Na definição dos elementos *codigo* e *nome* apenas será mostrado os conteúdos existentes em seus tags no documento XML. Na definição dos elementos *pre-requisito* e *ementa* os respectivos conteúdos serão colocados entre os tags da linguagem HTML <*div*>, <*b*> alterando a fonte para negrito ("bold") e <*font*> alterando a cor da fonte que será utilizada para escrever os literais *"Ementa:"* e *"Pré-Req.:"* Em seguida serão introduzidos os conteúdos dos elementos *pre-requisito* e *ementa* no documento HTML.

4.3.3 Especificação XSL de uma Ementa do Catálogo de Cursos

A seguir será utilizada a linguagem de estilo XSL que também pode ser utilizada na publicação de documentos nos formatos HTML e PS.

Para realizar o processamento dos documentos XML usando folhas de estilos XSL utilizamos a ferramenta *UFO* (Unicorn Formatting Objects) [32]. A ferramenta *UFO* utiliza a linguagem XSL versão 1.0, e está disponível para a plataforma Windows e de licensa *freeware*. Através dela é possível a criação de documentos no formato HTML e TeX.

A ferramenta *UFO* possui os seguintes módulos funcionais:

- **Transformações XSLT**: usado para transformações de certas estruturas de documentos XML para outras estruturas no formato XML, mudando os elementos existentes, a hierarquia entre os elementos, etc.
- **XSL-FO para HTML**: usado para a criação de documentos HTML a partir de documentos XML.
- **XSL-FO para Tex**: usado para a criação de documentos no formato TeX e posteriormente sua tradução para os formatos PS ou PDF, a partir de documentos XML.

Diferente da ferramenta Jade, UFO utiliza folhas de estilo XSL, as folhas de estilo XSL seguem as regras de formatação de um documento XML fazendo uso de tags e atributo em sua definição como mostra a figura 4.27, tornando a definição de folhas de estilo XSL mais complexa comparada à linguagem DSSSL (figura 4.25) utilizada pela ferramenta Jade, que se parece até com uma linguagem de programação funcional como Lisp sendo mais simplificada que a XSL. Além da simplicidade da linguagem de estilo DSSSL, ela também pode ser utilizada em documentos SGML, diferente de XSL que foi definida especificamente para o padrão XML.

A figura 4.27 é um exemplo de uma folha de estilo XSL utilizando XSL-FO para a disciplina do nosso exemplo (figura 4.1), obtendo o documento no formato PS representado pela figura 4.28.

Na figura 4.27 inicialmente são definidas informações sobre a folha de estilo através do elemento *xsl:stylesheet* que conterá as definições realizadas para a disciplina. O tipo do documento de saída é definido através do elemento *xsl:output*. O elemento *<xsl:template match="/">* define a formatação das páginas no documento de saída, como por exemplo o tamanho das margens. Através do elemento *<xsl:apply-templates>* serão processados os demais elementos do documento XML de entrada. O elemento *disciplina* utiliza as definições feitas anteriormente: fonte *Times* com um tamanho de *14pt* e com alinhamento justificado definido através do elemento *fo:block*, em seguida é construído outro bloco em negrito e centralizado para o cabeçalho *"Ementas do Catálogo de Cursos"*; o próximo bloco utiliza negrito e realiza chamadas para os elementos *codigo* e *nome*, onde são exibidos os conteúdos de cada elemento; mais dois blocos são construídos para os elementos *pre-requisito* e *ementa*, cujos conteúdos são exibidos depois de imprimirem os literais *"Pré-Req.:"* e *"Ementa:"* em negrito.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:uxt="http://www.unicorn-enterprises.com/UXT">
  <xsl:output method="uxt:fo-tex" indent="yes" encoding="UTF-8"/>
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master page-master-name="page"
          page-width="auto" page-height="auto">
          <fo:region-body margin-top="20mm" margin-bottom="45mm"
            margin-left="10mm" margin-right="10mm"/>
          <fo:region-before region-name="head-first" extent="20mm"/>
          <fo:region-after region-name="foot-first" extent="30mm"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <xsl:apply-templates/>
    </fo:root>
  </xsl:template>
  <xsl:template match="disciplina">
    <fo:page-sequence master-reference="page">
      <fo:flow flow-name="xsl-region-body">
        <fo:block font-family="Times" font-size="14pt"
          text-align="justify" padding="1cm">
          <fo:block font-weight="bold" text-align="center">
            Ementas do Catálogo de Cursos</fo:block>
          <fo:inline font-weight="bold">
            <xsl:apply-templates select="codigo"/>
            <xsl:apply-templates select="nome"/>
          </fo:inline>
          <fo:block> <xsl:apply-templates select="pre-requisito"/> </fo:block>
          <fo:block> <xsl:apply-templates select="ementa"/> </fo:block>
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
  </xsl:template>
  <xsl:template match="codigo"> <xsl:apply-templates/> </xsl:template>
  <xsl:template match="nome"> <xsl:apply-templates/> </xsl:template>
  <xsl:template match="pre-requisito">
    <fo:inline font-weight="bold">Pré-req.: </fo:inline> <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="ementa">
    <fo:inline font-weight="bold">Ementa: </fo:inline> <xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

Figura 4.27: Exemplo de uma folha de estilo XSL para uma disciplina no formato PS.

A figura 4.28 mostra a visualização de um documento no formato PS como resultado final do processamento utilizando a folha de estilo da figura 4.27.

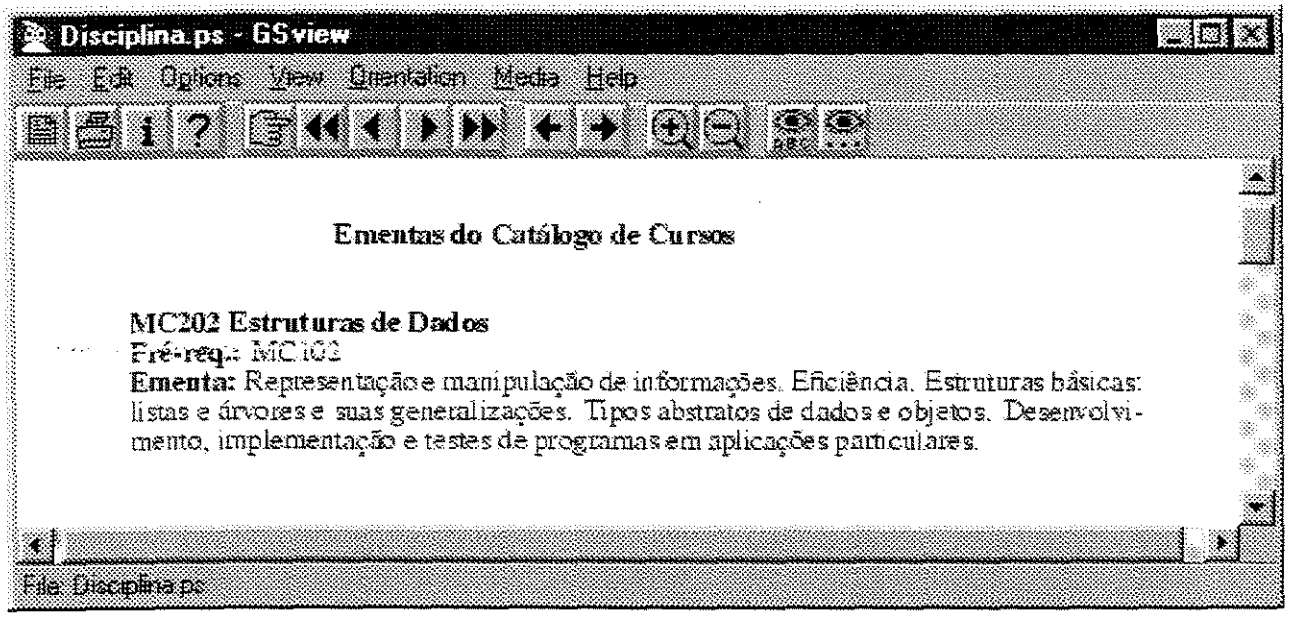

Figura 4.28: Exemplo de visualização de um documento PS utilizando uma folha de estilo XSL.

A figura 4.29 é um exemplo de uma folha de estilo XSL utilizando XSLT para obter um documento no formato HTML representado pela figura 4.30.

Na figura 4.29 inicialmente é definido o elemento *xsl:stylesheet*. O tipo do documento de saída é definido através do elemento *xsl:output* pelo par atributo/valor *method="html"*. O elemento <*xsl:template match="disciplina"*> define o cabeçalho e o título do documento HTML de saída, como também a formatação do corpo do documento utilizando diretamente os elementos presentes na linguagem HTML; também é especificado o uso da fonte *Arial,Helvetica*, a descrição do sub-título *"Ementas do Catálogo de Cursos"* no documento HTML, alterando a cor de sua fonte e utilizando negrito. Chamadas para os elementos *codigo* e *nome* são feitas para exibir os conteúdos de cada elemento com alinhamento à direita e em negrito; em seguida é inserida uma linha da cor azul através da imagem *linhazul.jpg*; adiante são inseridos os conteúdos dos elementos *pre-requisito* e *ementa*, através do comando <*xsl:apply-*

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
  result-ns="http://www.w3.org/TR/REC-html40">
  <xsl:output method="html" version="4.0"
    doctype-public="-//W3C//DTD HTML 4.0 Transitional//EN"
    doctype-system="http://www.w3.org/TR/REC-html40/loose.dtd"
    encoding="ISO-8859-1" />
    <xsl:template match="disciplina">
      <html>
        <head>
          <title>UNICAMP - Ementas das Disciplinas do Catálogo de Cursos</title>
        </head>
        <body>
          <center><table cols="1" width="600"><tr><td>
            <font face="Arial,Helvetica" size="-1">
            <center><br><b><font color="#000099">
              Ementas do Catálogo de Cursos</font></b></br></center>
            <div align="right"><br><b><font color="#000099">
              <xsl:apply-templates select="codigo" />
              <xsl:apply-templates select="nome" />
            </font></b></br><img SRC="./linhazul.jpg" /></div>
            <div align="left"><br>
              <xsl:apply-templates select="pre-requisito" />
              <xsl:apply-templates select="ementa" />
            </br></div></font>
          </td></tr></table></center>
        </body>
      </html>
    </xsl:template>
    <xsl:template match="codigo"> <xsl:apply-templates/> </xsl:template>
    <xsl:template match="nome"> <xsl:apply-templates/> </xsl:template>
    <xsl:template match="pre-requisito">
      <b><font color="#000099">Pré-Req.: </font></b>
      <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="ementa">
      <b><font color="#000099">Ementa: </font></b>
      <xsl:apply-templates/>
    </xsl:template>
</xsl:stylesheet>
```

Figura 4.29: Exemplo de uma folha de estilo XSL para uma disciplina no formato HTML.

templates/> depois de imprimirem os literais *"Pré-Req.:"* e *"Ementa:"* em negrito e com alteração na cor de sua fonte.

A figura 4.30 mostra a visualização do documento HTML como resultado final do processamento utilizando a folha de estilo da figura 4.29.

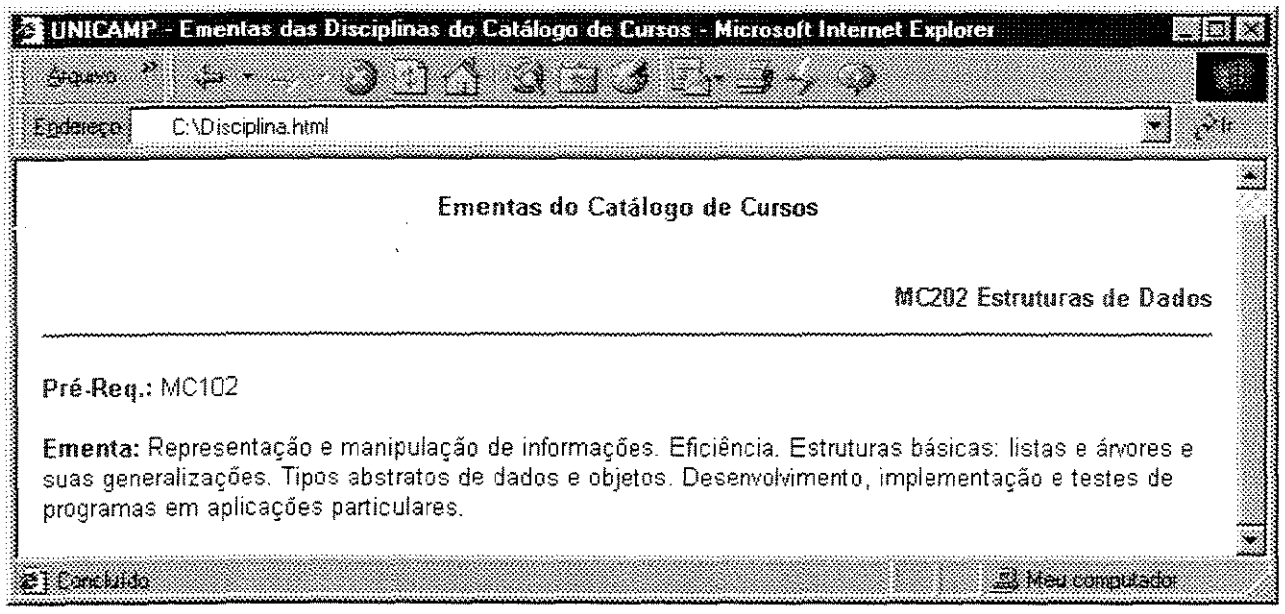

Figura 4.30: Exemplo de visualização de um documento HTML utilizando uma folha de estilo XSL.

Capítulo 5

Resultados Obtidos

Nesse capítulo serão apresentados os resultados obtidos pela aplicação da metodologia desenvolvida para a elaboração dos catálogos dos cursos de graduação e pós-graduação da Unicamp.

A primeira seção descreve o processamento realizado no catálogo de pós-graduação do Instituto de Computação (IC) da Unicamp e exibe parte do documento final resultante. Na seção posterior é descrito o processamento feito no catálogo de graduação da Unicamp.

5.1 Catálogo da Pós-Graduação do IC

Foi analisada a organização do catálogo de pós-graduação do IC, desde a estruturação da capa introdutória, organização dos departamentos e comissões, informações sobre o corpo docente, descrição dos cursos, normas específicas e linhas de pesquisa, e finalmente a descrição das disciplinas oferecidas nos cursos da pós-graduação do Instituto de Computação da Unicamp.

Um conjunto de documentos XML foi construído a partir dessa análise utilizando um documento DTD para a definição dos elementos e sua organização hierárquica, com todas as características presentes no catálogo.

Foi construída uma folha de estilo utilizando a linguagem DSSSL definindo a formatação final do catálogo, incluindo a utilização de duas colunas em algumas áreas, o tipo e o tamanho da fonte, a utilização de negrito e itálico, a definição de cabeçalhos e rodapés e a ordem de apresentação dos elementos XML.

Em seguida é mostrado o documento final no formato PS obtido a partir do processamento dos documentos XML, respectivos DTDs, folhas de estilo DSSSL, utilizando o processador *Jade* (James DSSSL Engine). Partes dos documentos XML e DTD relacionados ao catálogo da pós-graduação estão no apêndice.

GOVERNO DO ESTADO DE SÃO PAULO
UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE COMPUTAÇÃO

CATÁLOGO DOS CURSOS

DE PÓS-GRADUAÇÃO

2001

INSTITUTO DE COMPUTAÇÃO

Diretor: **Tomasz Kowaltowski**
Diretor Associado: **Ricardo de Oliveira Anido**
Secretária: **Maria Magali Chaguri**

DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

Tomasz Kowaltowski, MS-6, RDIDP
Célio Cardoso Guimarães, MS-5, RDIDP
Edmundo Roberto Mauro Madeira, MS-4, RDIDP
Nelson Luís Saldanha da Fonseca, MS-4, RDIDP
Paulo Lício de Geus, MS-4, RDIDP
Ricardo de Oliveira Anido, MS-4, RDIDP
Rogério Drummond Burnier Pessoa de Mello Filho, MS-4, RDIDP
Alexandre Xavier Falcão, MS-3, RDIDP
Arthur João Catto, MS-3, RTC
Guido Costa Souza de Araújo, MS-3, RDIDP
Maria Beatriz Felgar de Toledo, MS-3, RDIDP
Mario Lúcio Côrtes, MS-3, RTC
Paulo César Centoducatte, MS-3, RDIDP
Ricardo Pannain, MS-3, RTP
Fernando Antonio Vanini, MS-2, RTP
Silvia Helena Machado de Oliveira, MS-2, RDIDP

DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

Claudia Maria Bauzer Medeiros, MS-5, RDIDP
Ariadne Maria Brito Rizzoni Carvalho, MS-4, RDIDP
Geovane Cayres Magalhães, MS-4, RTP
Hans Kurt Edmund Liesenberg, MS-4, RDIDP
Heloisa Vieira da Rocha, MS-4, RDIDP
Jacques Wainer, MS-4, RDIDP
Luiz Eduardo Buzato, MS-4, RDIDP
Maria Cecília Calani Baranauskas, MS-4, RDIDP
Neucimar Jerônimo Leite, MS-4, RDIDP
Cecília Mary Fischer Rubira, MS-3, RDIDP
Eliane Martins, MS-3, RDIDP
Thelma Cecília dos Santos Chiossi, MS-2, RTC
Sindo Vasquez Dias, MS-1, RTP

DEPARTAMENTO DE TEORIA DA COMPUTAÇÃO

Cláudio Leonardo Lucchesi, MS-6, RDIDP
Jorge Stolfi, MS-5, RDIDP
Arnaldo Vieira Moura, MS-4, RDIDP
Célia Picinin de Mello, MS-4, RDIDP
Cid Carvalho de Souza, MS-4, RDIDP
João Carlos Setubal, MS-4, RDIDP
João Meidanis, MS-4, RDIDP
Pedro Jussieu de Rezende, MS-4, RDIDP
Anamaria Gomide, MS-3, RDIDP
Flávio Keidi Miyazawa, MS-3, RDIDP
Ricardo Dahab, MS-3, RDIDP

COMISSÃO DE PÓS-GRADUAÇÃO

Pedro Jussieu de Rezende, *Coordenador*
Luiz Eduardo Buzato, *Membro*
Maria Cecília Calani Baranauskas, *Membro*
Adilson Luiz Bonifácio, *Suplente*
Flávio Keidi Miyazawa, *Suplente*
Sandro Rigo, *Representante Discente*

• INTRODUÇÃO

O Programa de Pós-graduação em Computação, teve início em março de 1977, no Instituto de Matemática, Estatística e Computação Científica (IMECC) e passou a funcionar junto ao Instituto de Computação a partir da criação deste pela Deliberação Consu-A-1, de 26 de março de 1996.

Estão em funcionamento os Cursos de Mestrado e Doutorado em Computação.

Várias informações atualizadas estão disponíveis na web: http://www.dcc.unicamp.br/cpg

• CORPO DOCENTE

Professores Plenos

Alexandre Xavier Falcão, *Eng. Elétr. (UFPe, PE, 1988); Mestre (Unicamp, 1993); Doutor (Unicamp, 1996).*
Anamaria Gomide, *Bach. Matemática (UnB, 1974); Mestre (Unicamp, 1999).*
Ariadne Maria Brito Rizzoni Carvalho, *Anal. Sistemas (PUC-CAMP, 1979); Doutor (Reading, 1989); Livre Docente.*
Arnaldo Vieira Moura, *Eng. Elétr. (ITA, 1973); Mestre (ITA, 1976); Doutor (Univ. of California, Berkeley, 1980); Livre Docente.*
Arthur João Catto, *Eng. Civil (USP, São Carlos, 1970); Mestre (USP, São Carlos, 1977); Mestre (Manchester Univ., 1978); Doutor (Manchester Univ., 1981).*
Cecília Mary Fischer Rubira, *Bach. Ciên. Comp. (Unicamp, 1985); Mestre (Unicamp, 1990); Doutor (Univ. of Newcastle upon Tyne, 1994).*
Célia Picinin de Mello, *Lic. Mat. (FFCL, Rio Claro, 1974); Mestre (Unicamp, 1984); Doutor (UFRJ, 1992).*
Célio Cardoso Guimarães, *Eng. Elétr. (ITA, 1965); Mestre (Case Western Reserve Univ., Cleveland, 1970); Doutor (Case Western Reserve Univ., Cleveland, 1973).*
Cid Carvalho de Souza, *Eng. Elétr. (PUC, RJ, 1985); Mestre (PUC, RJ, 1989); Doutor (Univ. Catholique de Louvain, Bélgica, 1993); Livre Docente.*
Claudia Maria Bauzer Medeiros, *Eng. Elétr. (PUC, RJ, 1976); Mestre (PUC, RJ, 1979); Doutor (Univ. Waterloo, Canadá, 1985); Livre Docente (Unicamp, 1992).*
Cláudio Leonardo Lucchesi, *Eng. Elétr. (USP, 1969); Mestre (Univ. Waterloo, Canadá, 1970); Doutor (Univ. Waterloo, Canadá, 1976).*
Edmundo Roberto Mauro Madeira, *Eng. Civil (Unicamp, 1980); Mestre (Unicamp, 1985); Doutor (Unicamp, 1991).*
Eliane Martins, *Bach. Inf. (UFRJ, 1977); Mestre (COPPE, UFRJ, 1993); Doutor (École Nationale Supérieure de L'Aéronautique et de L'Espace, Toulouse, 1992).*

Flávio Keidi Miyazawa, *Bach. Ciên. Comp. (UFMS, 1990); Mestre (USP, 1993); Doutor (USP, 1997).*
Geovane Cayres Magalhães, *Eng. Civil (UFBa, 1971); Mestre (PUC, RJ, 1976); Doutor (Univ. de Toronto, 1981); Livre Docente.*
Guido Costa Souza de Araújo, *Eng. Elétr. (UFPe, 1985); Mestre (Unicamp, 1991); Mestre (Princeton Univ., 1994); Doutor (Princeton Univ., 1997).*
Hans Kurt Edmund Liesenberg, *Bach. Ciên. Comp. (Unicamp, 1976); Mestre (Unicamp, 1980); Doutor (Univ. Newcastle Upon Tyne, 1985).*
Heloisa Vieira da Rocha, *Bach. Ciên. Comp. (Unicamp, 1975); Mestre (Unicamp, 1981); Doutor (Unicamp, 1991).*
Jacques Wainer, *Eng. Elétr. (USP, 1982); Doutor (Pensylvania State Univ., 1991); Livre Docente.*
João Carlos Setubal, *Eng. Mec. (USP, 1979); Mestre (Unicamp, 1987); Doutor (Univ. Washington, 1992); Livre Docente.*
João Meidanis, *Bach. Mat. (USP, 1980); Mestre (USP, 1984); Mestre (Univ. Wisconsin, 1989); Doutor (Univ. Wisconsin, 1992); Livre Docente (Unicamp, 1996).*
Jorge Stolfi, *Eng. Elétr. (USP, 1973); Mestre (USP, 1979); Doutor (Stanford Univ., 1989); Livre Docente (Unicamp, 1996).*
Luiz Eduardo Buzato, *Bach. Ciên. Comp. (Unicamp, 1985); Mestre (Unicamp, 1990); Doutor (Univ. of New Castle Upon Tyne, UK, 1994).*
Maria Beatriz Felgar de Toledo, *Bach. Ciên. Comp. (Unicamp, 1980); Mestre (Unicamp, 1986); Doutor (Univ. Lancaster, 1992).*
Maria Cecília Calani Baranauskas, *Bach. Ciên. Comp. (Unicamp, 1976); Mestre (Unicamp, 1981); Doutor (Unicamp, 1993).*
Mário Lúcio Côrtes, *Eng. Elétr. (ITA, 1973); Mestre (USP, 1980); Doutor (Stanford, 1987); Livre Docente.*
Nelson Luís Saldanha da Fonseca, *Eng. Elétr. (PUC, RJ, 1984); Mestre (Univ. Southern California, 1993); Doutor (Univ. Southern California, 1994); Livre Docente.*
Neucimar Jerônimo Leite, *Eng. Elétr. (UFPB, 1985); Mestre (UFPB, 1988); Doutor (Univ. Paris VI, 1993).*
Paulo Cesar Centoducatte, *Eng. Elétr. (UFES, 1982); Mestre (Unicamp, 1992); Doutor (Unicamp, 2000).*
Paulo Lício de Geus, *Eng. Elétr. (ITA, 1976); Mestre (Unicamp, 1979); Doutor (Univ. Manchester, 1985).*
Pedro Jussieu de Rezende, *Bach. Mat. (Univ. Brasília, 1977); Mestre (Unicamp, 1979); Doutor (Northwestern Univ., 1988); Livre Docente (Unicamp, 1996).*
Ricardo Dahab, *Bach. Ciên. Comp. (Unicamp, 1978); Mestre (Unicamp, 1984); Doutor (Waterloo, 1993).*
Ricardo de Oliveira Anido, *Eng. (ITA, 1978); Mestre (Unicamp, 1984); Doutor (Imperial College, 1989).*
Ricardo Pannain, *Eng. Elétr. (FEEC, 1982); Mestre (Unicamp, 1988); Doutor (Unicamp, 1999).*
Rogério Drummond Burnier Pessoa de Mello Filho, *Bach. Ciên. Comp. (Unicamp, 1978); Mestre (Unicamp, 1980); Doutor (Cornell Univ., 1985).*
Tomasz Kowaltowski, *Eng. Elétr. (USP, 1966); Mestre (Univ. California, Berkeley, 1970); Doutor (Univ. California, Berkeley, 1980); Livre Docente (Unicamp).*

• DESCRIÇÃO DOS CURSOS

Credenciamento:

O Programa de Pós-Graduação em Computação, no Processo de Avaliação da CAPES, referente ao Biênio 1997/1998, recebeu conceito "5" (Equivalente ao conceito "A" nas avaliações de anos anteriores).

Integralização:

O curso de Mestrado em Computação será integralizado em um mínimo de 12 meses e em um máximo de 36 meses.

O curso de Doutorado em Computação será integralizado em um mínimo de 24 meses e em um máximo de 60 meses.

As disciplinas estão agrupadas em introdutórias, básicas, especializadas, tópicos especiais e seminários.

Disciplinas do Programa:

Disciplinas Introdutória:

MO203 180 12 Conceitos Básicos em Ciência da Computação

OBSERVAÇÃO: Esta disciplina não tem seus créditos computados para efeito de conclusão do programa.

Disciplinas Básicas:

MO401 180 12 Arquitetura de Computadores I
MO403 180 12 Implementação de Linguagens I
MO405 180 12 Teoria dos Grafos I
MO406 180 12 Linguagens Formais e Autômatos
MO409 180 12 Engenharia de Software I
MO410 180 12 Bancos de Dados
MO416 180 12 Introdução à Inteligência Artificial
MO417 180 12 Complexidade de Algoritmos I
MO441 180 12 Computação Distribuída
MO442 180 12 Processamento e Análise de Imagens

Disciplinas Especializadas:

MO601 180 12 Arquitetura de Computadores II
MO603 180 12 Computação Gráfica
MO611 180 12 Teleprocessamento e Redes
MO614 180 12 Computabilidade e Funções Recursivas
MO615 180 12 Implementação de Linguagens II
MO617 180 12 Sistemas Operacionais Distribuídos
MO618 180 12 Teste de Circuitos Digitais
MO619 180 12 Geometria Computacional
MO620 180 12 Engenharia de Software II
MO622 180 12 Fatores Humanos em Sistemas de Computação
MO625 180 12 Processamento de Linguagem Natural
MO626 180 12 Representação de Conhecimento e Raciocínio
MO633 180 12 Bancos de Dados II
MO637 180 12 Complexidade de Algoritmos II
MO638 180 12 Administração de Redes de Computadores
MO639 180 12 Segurança de Redes de Computadores
MO640 180 12 Biologia Computacional
MO641 180 12 Projeto e Implementação de Sistemas Distribuídos
MO642 180 12 Inteligência Artificial e Educação
MO645 180 12 Projeto de Interfaces de Usuário
MO646 180 12 Construção de Interfaces de Usuário
MO647 180 12 Introdução ao Projeto de Sistemas VLSI

MO648 180 12 Projeto de Redes Multimídia
MO650 0 72 Tese de Mestrado
MO669 90 06 Estudo Dirigido II
MO800 0 144 Tese de Doutorado

Tópicos Especiais:

MO801 180 12 Tópicos em Arquitetura e Hardware
MO802 180 12 Tópicos em Linguagens de Programação
MO804 180 12 Tópicos em Teoria dos Grafos
MO805 180 12 Tópicos em Recuperação de Informação
MO806 180 12 Tópicos em Sistemas Operacionais
MO809 180 12 Tópicos em Computação Distribuída
MO810 180 12 Tópicos em Inteligência Artificial
MO812 180 12 Tópicos em Bancos de Dados
MO814 180 12 Tópicos em Computação Gráfica
MO815 180 12 Tópicos em Processamento de Imagens
MO817 180 12 Tópicos em Semântica e Verificação de Programas
MO818 180 12 Tópicos em Redes de Computadores I
MO821 180 12 Tópicos em Redes de Computadores II
MO823 180 12 Tópicos em Complexidade de Algoritmos
MO824 180 12 Tópicos em Otimização Combinatória
MO827 180 12 Tópicos em Engenharia de Software I
MO828 180 12 Tópicos em Engenharia de Software II
MO829 180 12 Tópicos em Teoria de Computação

Seminários:

MO901 45 03 Seminário de Computação

• LINHAS DE PESQUISA

As Dissertações e Teses poderão ser desenvolvidas principalmente nas seguintes áreas: algoritmos paralelos, ambientes e paradigmas de programação, análise de algoritmos, arquitetura de computadores, bancos de dados, biologia computacional, computação gráfica, métodos numéricos, criptografia computacional, engenharia de software, geometria computacional, groupware, informática na educação, inteligência artificial, interfaces homem-computador, linguagens de programação e compiladores, linguagens formais e autômatos, otimização combinatória, processamento de imagens, processamento de linguagens naturais, projeto e testes de sistemas digitais, redes de computadores, sistemas de informações geográficas, sistemas distribuídos, sistemas operacionais, sistemas tolerantes a falhas, teoria dos grafos.

• NORMAS ESPECÍFICAS

Admissão

Os candidatos interessados deverão encaminhar à Secretaria de Pós-graduação do IC ficha de inscrição devidamente preenchida, além dos documentos exigidos conforme o programa específico, entre os quais estão: históricos escolares; cartas de recomendação e curriculum vitae resumido.

As inscrições são aceitas até 31 de outubro para os candidatos que pretendem iniciar o programa em janeiro ou março seguinte. Os candidatos serão notificados sobre a aceitação ou não até o dia 15 de dezembro. Excepcionalmente poderão ser aceitos candidatos para iniciarem o programa em agosto. Neste caso, o prazo de inscrição encerra-se em 31 de maio e os candidatos serão notificados sobre a aceitação ou não até o dia 15 de julho. Poderá ocorrer o fato de a aceitação do candidato ao Mestrado ficar condicionada ao bom rendimento obtido em disciplinas introdutórias oferecidas em um curso de verão nos meses de janeiro/fevereiro ou ser aprovado em exame de admissão. O título de Mestre obtido no Instituto ou fora dele não dá ao candidato o direito de matricular-se automaticamente no Doutorado. Em casos excepcionais, candidatos sem o título de Mestre poderão ser aceitos para o Doutorado. Poderão ser aceitos alunos para cursar disciplinas isoladas. Os alunos nessas condições serão "Estudantes Especiais da Unicamp" e os créditos assim obtidos poderão ser convalidados no caso daqueles que se tornarem alunos regulares na Pós-graduação do IC. Inscrições para este fim devem ser feitas antecipadamente na secretaria, conforme calendário próprio do IC.

Bolsas de Estudo

O IC dispõe de um numero limitado de bolsas de estudo da CAPES e do CNPq. Estas bolsas são fornecidas aos alunos de Mestrado ou Doutorado aceitos, segundo critérios acadêmicos. Este tipo de bolsa também poderá ser solicitada diretamente pelo aluno e seu orientador à FAPESP.

Esquema dos Cursos

A Comissão de Pós-graduação (CPG) indicará para cada aluno de Mestrado ou Doutorado um orientador inicial, que poderá continuar como Orientador de Dissertação ou Tese ou poderá ser substituído para esse fim por outro Docente. Este orientador inicial fixará, para cada candidato e de acordo com este, seu programa de estudos.

Requisito para Obtenção dos Títulos

Mestrado em Computação

Os candidatos ao Mestrado devem ter concluído ou estar concluindo um curso de graduação, preferencialmente em Computação, Engenharia ou Matemática.

Para obter o grau de Mestre em Computação pelo Instituto, o aluno deverá:

1 - Conseguir, até 12 meses após o ingresso, a aprovação em disciplinas de Pós-graduação do Instituto, totalizando pelo menos 72 créditos, assim distribuídos:

Uma disciplina da área de Teoria da Computação, obrigatoriamente no primeiro semestre após o ingresso no curso - 12 créditos;

Uma disciplina na área de Sistemas de Computação - 12 créditos;

Uma disciplina na área de Sistemas de Programação - 12 créditos;

Outras disciplinas, à escolha do aluno e com aprovação do orientador - 24 créditos;

Dois semestres da disciplina Seminário de Computação - 6 créditos;

Uma disciplina de Estudo Dirigido entre o primeiro e segundo semestre do aluno no curso - 6 créditos;

2 - Definir, ao fim do Estudo Dirigido (cerca de 6 meses após o início no curso), o tema da dissertação; escrever um plano de trabalho (10-20 páginas) para a mesma; apresentar esse plano oralmente durante o Exame de Qualificação de Mestrado, e obter a aprovação desta última;

3 - Demonstrar capacidade de compreensão de texto técnico em inglês, em exame ministrado pelo Instituto, dentro do primeiro ano do aluno no curso;

4 - Completar e entregar a dissertação de Mestrado preferencialmente até 24 meses após o início do programa, apresentá-la oralmente perante uma banca de pelo menos três docentes (incluindo o orientador e um membro externo ao Instituto) e obter a aprovação desta.

Durante o transcorrer do primeiro semestre do aluno no curso, o aluno deverá escolher um orientador definitivo: um docente do Instituto que esteja disposto a supervisionar seu Estudo Dirigido, suas pesquisas no restante do curso e a elaboração da Dissertação.

Doutorado em Computação

Os candidatos ao Doutorado devem ter concluído um curso de Pós-graduação, de bom nível, em Computação, Engenharia ou Matemática. Em casos excepcionais poderão ser admitidos candidatos graduados, preferencialmente, em Computação, Engenharia ou Matemática sem o título de Mestre.

Além disso, um candidato ao curso de Doutorado somente será aceito se houver um ou mais docentes do Instituto que estejam dispostos a orientá-lo na sua área de interesse. Uma vez que a CPG estabelece limites para o número de orientandos por docentes, recomenda-se fortemente que candidatos ao Doutorado entrem em contato com possíveis orientadores, para confirmar sua disponibilidade e interesse, antes de se inscreverem para o programa.

Para obter o grau de Doutor em Computação pelo Instituto, o aluno deverá:

1 - Obter, até 24 meses após o ingresso no curso, a aprovação em disciplinas de Pós-graduação aprovadas pela CPG, totalizando pelo menos 72 créditos.

2 - Ser aprovado, no prazo estabelecido no Regulamento do Curso, nos exames de qualificação geral do Instituto, em pelo menos três das seguintes áreas:

- Teoria da Computação (obrigatória);
- Sistemas de Informação;
- Sistemas de Programação;
- Sistemas de Computação;

3 - Ser aprovado, até no máximo 12 meses após o exame de qualificação geral, no exame de qualificação específico sobre o assunto da tese, administrado por uma banca indicada pela CPG;

4 - Demonstrar capacidade de compreensão de texto técnico em inglês em exame ministrado pelo Instituto dentro do primeiro ano do aluno no curso;

5 - Completar e entregar uma tese sobre resultados originais de pesquisa, defendê-la oralmente perante uma banca julgadora de pelo menos cinco docentes (incluindo o orientador e 2 membros externos ao Instituto), e obter a aprovação desta.

• INSTALAÇÕES E EQUIPAMENTOS

O campus dispõe de uma rede de fibra ótica, que interliga os equipamentos da Unicamp e as redes nacionais e internacionais, via Internet. A ligação do IC a essas redes se dá através da Rede ANSP (Academic Network of São Paulo), gerenciada pela FAPESP. O Instituto conta com uma grande variedade de equipamentos computacionais. São diversos tipos de estações de trabalho SUN (Enterprise, Ultra, SPARC 1000, 20, 10, 4), PC Pentium, Macintosches e Silicon Graphics. Recentemente, adquiriu, através de projeto de pesquisa financiado pela FAPESP, computadores de alto desempenho de arquiteturas variadas, que vão desde máquinas multiprocessadas de memória compartilhada até máquinas paralelas do tipo "Beowulf". Alem destes equipamentos, o IC dispõe de um bom numero de impressoras laser e de scanners de mesa.

Algumas das pesquisas em andamento fazem uso dos computadores (IBM SP 2) do CENAPAD (Centro Nacional de Processamento de Alto Desempenho) que se encontra situado dentro do campus.

• IDENTIFICAÇÃO DAS DISCIPLINAS

• LEGENDA

As disciplinas oferecidas pela unidade encontram-se a seguir identificadas. As informações são, na ordem em que aparecem, as seguintes:

- Código da Disciplina
- Nome da disciplina
- T - Total de horas aulas teóricas.
- E - Total de horas aulas de exercícios.
- L - Total de horas de laboratório ou de seminários.
- S - Total de horas de trabalho de campo.
- I - Total de horas de estudo dirigido ou de estudo em casa.
- C - Total de créditos. Cada crédito corresponde a 15 (quinze) horas de atividades.
- P - Período mais provável da oferta da disciplina, de acordo com a convenção:
  - 1 - 1º período letivo
  - 2 - 2º período letivo
  - 3 - qualquer período letivo
- Os pré-requisitos (PR): exigidos para a matrícula na disciplina. AA200 - Significa Autorização da respectiva CPG.
- A ementa descreve sucintamente o assunto relacionado com a disciplina. Em algumas disciplinas, principalmente aquelas relacionadas com Tópicos Especiais, as ementas serão oferecidas pelas Unidades de Ensino correspondentes, na época da oferta dessas disciplinas.
- O livro onde se encontra o material básico (texto) pode também constar da informação de cada disciplina. No caso do material se encontrar em várias fontes, a lista bibliográfica será oportunamente fornecida pelo Professor Responsável pela disciplina.

• EMENTAS DAS DISCIPLINAS

MO203 Conceitos Básicos em Ciência da Computação

T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Fundamentos matemáticos em análise de algoritmos (somas, probabilidade, notação assintótica e indução matemática). Estruturas de dados elementares. Algoritmos de

Busca, ordenação e estatísticas de ordem. Grafos (representação de grafos e algoritmos básicos: buscas e ordenação topológica). Fundamentos de projeto. Processador básico. Conjunto de instruções. Pipelining. Hierarquia da memória. Caches.
Bibliografia: Introduction to Algorithms. Cormen, T.H., Leiserson, C.E., Rivest, R.L. McGraw Hill, 1990. Introduction to Algorithms, a Creative Approach. Manber, U. Addison-Wesley, 1989. Computer Organization and Design: The Hardware/Software Interface. David A. Patterson e John L. Hennessy, Morgan Kaufmann Publishers, 1994.

MO401 Arquitetura de Computadores I
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Uma introdução avançada a arquitetura e organização de computadores. Tecnologias e perspectiva histórica. Medidas de desempenho. Conjunto de instruções. Unidades de aritmética e lógica. Projeto básico de um processador. Pipeline. Hierarquia da memória: cache e memória virtual. Dispositivos de I/O. Processamento paralelo.
Bibliografia: David A. Patterson and John L. Hennessy. Computer Organization and Design: The Hardware/Software Interface, 1997, 2nd Edition. Morgan Kauffman. John L. Hennessy and David A. Patterson, Morgan Kauffman. Computer Architecture: A Quantitative Approach, 1996, 2nd Edition. Morgan Kauffman.

MO403 Implementação de Linguagens I
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Descrição formal de linguagens de programação. Análise léxica. Análise sintática. Geração de código. Sistemas de execução: blocos, procedimentos, recursão. Recuperação de erros. Ferramentas para construção de analisadores léxicos, sintáticos e semânticos. Construção de um compilador para uma linguagem exemplo.
Bibliografia: Kowaltowski, T., Implementação de Linguagens de Programação, Editora Guanabara Dois, 1983. Aho A. V., Sethi R. e Ullman, J. D., Compilers - Principles, Techniques, and Tools, Addison-Wesley, 1986. Schreiner, A. T. e Friedman Jr., H.G. Introduction to Compiler Construction With UNIX, Pretince-Hall, 1985, Andrew W. Appel, Modern Compiler Implementation in Java, Cambridge University Press, 1998.

MO405 Teoria dos Grafos I
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Grafos, subgrafos, grafos orientados, famílias de grafos. árvores, caminhos, ciclos. Conexidade. Grafos eulerianos. Grafos hamiltonianos. Emparelhamento em grafos bipartidos. Coloração de arestas. Coloração de vértices. Conjuntos independentes. Teoria de Ramsey. Grafos planares.
Bibliografia: R. Diestel em Graph Theory. Springer-Verlag, 1997 A. Bondy, e U. S. R. Murty. em Graph Theory with Applications. North-Holland, 1976.

MO406 Linguagens Formais e Autômatos
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Palavras e linguagens. Gramáticas regulares e autômatos finitos. Não determinismo e generalizaçSes. Minimização de estados. Expressões regulares. Teorema da interação para linguagens regulares. Gramáticas livre de contexto e autômatons da pilha. Determinismo e ambigüidade. Teorema da interação para linguagens livres de contexto. Gramáticas sensíveis ao contexto e autômatons linares. Gramáticas e máquinas de Turing. Generalizações e restrições. Determinismo e algoritmos. Recursão e enumeração. Decidibilidade. O problema de Post. OperaçSes com linguagens. Transdutores e operações fechadas.
Bibliografia: Hopcroft, J. e Ullman, J., "Introduction to Automata Theory, Languages, and Computation", Addison Wesley, 1979. Sipser, M "Introduction to the Theory of Computation", PWS Pub. Co., 1997. Kelley, D. "Automata and Formal Languages", Prentice Hall, 1995. Floyd, R. e Beigel, R. "The Language of Machines: An Introduction to Computability and Formal Languages", W. H. Freeman Co., 1994.

MO409 Engenharia de Software I
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Processos e modelos de desenvolvimento de software. Padrões de codificação. Teste e verificação de programas. Distribuição e manutenção de software. Metricas de complexidade de software.
Bibliografia: Pressman, R.S., "Software Engineering: A Practitioner's Approach", 4ª ed., McGraw Hill, 1997. Sommerville, I., "Software Engineering", 5ª ed., Addison Wesley, 1996. Ghezzi, C. Jazayeri, M. e Mandrioli, D., "Fundamentals of Software Engeneering", Prentice-Hall, 1991.

MO410 Bancos de Dados
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Introdução a sistemas de banco de dados incluindo modelos de dados, técnicas e teoria de projeto de bancos de dados, processamento de consultas e atualizações, esquemas para organizar e indexar arquivos e processamento de transações.
Bibliografia: Ullman, J. D. Principles of Database and Knowledge Base Systems, volumes I e II, Computer Science Press, 1988 e 1990. Elmasri R. e Navathe, S. Fundamentals of Database Systems. Benjamin Cummings, 1994.

MO416 Introdução à Inteligência Artificial
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Histórico. Representação de conhecimento. Busca de informação e teoria de jogos. Inteligência Artificial Distribuída. Conexionismo. Aplicações: resolução de problemas, aprendizagem, processamento de língua natural, visão, robótica, sistemas especialistas e agentes inteligentes.
Bibliografia: Russel. S e Norvig, P.; Artificial Intelligence: a modern approach, Prentice Hall, 1995. Winston, P. H., (1992) Artificial Intelligence, 3ª ed., Addison-Wesley. Firebaugh, M. W., (1988) Artificial Intelligence: Knowledge-Based Approach, Boyd and Fraser.

MO417 Complexidade de Algoritmos I
T:60 E:0 L:0 S:0 I:120 C:12 P:3
Pré-req.: AA220
Ementa: Modelos de computação e ferramentas/notação para análise de algoritmos. Indução matemática e projeto de algoritmos. Algoritmos gulosos. Programação dinâmica. Divisão e conquista. Algoritmos para ordenação e seleção. Algoritmos para problemas básicos em grafos. Reduções e NP-completude.
Bibliografia: Cormen, Leiserson e Rivest. Introduction to Algorithms, MIT Press, 1990. U. Manber. Introduction to Algorithms. Addison Wesley, 1989. Brassard and Bratley. Algorithms. Prentice-Hall, 1996. Garey and Johnson. Computers and Intractability. Freeman, 1982.

MO441 Computação Distribuída
T:60 E:0 L:0 S:0 I:120 C:12 P:3

5.2 Catálogo da Graduação

Foi analisada a organização do catálogo de cursos da graduação da Unicamp, desde a estruturação de uma capa introdutória, organização administrativa da universidade, o conselho universitário, comissão central de graduação, as atividades multidisciplinares e a descrição dos cursos de graduação oferecidos pela Unicamp, divididos por faculdades e institutos: seus respectivos cursos, organização dos responsáveis pelo oferecimento (diretores, coordenadores, etc), descrição do curso, disciplinas relacionadas, corpo docente, separados por departamento e finalmente a descrição de todas as disciplinas oferecidas.

Não foram utilizadas todas as informações existentes no catálogo de graduação devido à grande extensão do catálogo, sendo exemplificados apenas alguns cursos.

Um conjunto de documentos XML foi construído a partir na análise descrita nos parágrafos anteriores, utilizando um documento DTD para a definição dos elementos e sua organização hierárquica representando todas as características presentes no catálogo de cursos.

Com as informações do catálogo representadas em documentos XML e seguindo as regras definidas no respectivo DTD, foi construída uma folha de estilo DSSSL representando a visualização final desejada. A folha de estilo DSSSL define inclusive a área utilizada em cada página do catálogo, o tipo e o tamanho da fonte utilizada, assim como a utilização de negrito e itálico, a definição de cabeçalhos e rodapés e a ordem de apresentação dos elementos contidos nos documentos XML.

A seguir é mostrado o documento no formato PS obtido através do processamento dos documentos XML, seus respectivos DTDs, e a folha de estilo DSSSL utilizando o processador *Jade* (James DSSSL Engine).

GOVERNO DO ESTADO DE SÃO PAULO
UNIVERSIDADE ESTADUAL DE CAMPINAS

CATÁLOGO DOS CURSOS DE GRADUAÇÃO

2001

ADMINISTRAÇÃO SUPERIOR DA UNIVERSIDADE

GOVERNADOR DO ESTADO DE SÃO PAULO
MÁRIO COVAS

REITOR DA UNIVERSIDADE ESTADUAL DE CAMPINAS
HERMANO DE MEDEIROS FERREIRA TAVARES

COORDENADOR GERAL DA UNIVERSIDADE
FERNANDO GALEMBECK

PRÓ-REITOR DE EXTENSÃO E ASSUNTOS COMUNITÁRIOS
ROBERTO TEIXEIRA MENDES

PRÓ-REITOR DE DESENVOLVIMENTO UNIVERSITÁRIO
LUÍS CARLOS GUEDES PINTO

PRÓ-REITOR DE PESQUISAS
IVAN EMÍLIO CHAMBOULEYRON

PRÓ-REITOR DE GRADUAÇÃO
ANGELO LUIZ CORTELAZZO

PRÓ-REITOR DE PÓS-GRADUAÇÃO
JOSÉ CLÁUDIO GEROMEL

CONSELHO UNIVERSITÁRIO

Reitor
Hermano de Medeiros Ferreira Tavares

Secretário Geral
Paulo Solíero

Conselheiros

Adilson Roberto Cardoso	Representante dos Professores Assistentes
Águeda Bernardete Bittencourt	Diretor da Faculdade de Educação
Alcir José Monticelli	Representantes dos Professores Titulares
Angelo Luiz Cortelazzo	Pró-Reitor de Graduação
Antonio Carlos Gilli Martins	Representante dos Professores Assistentes
Adilson Roberto Cardoso	Representante dos Professores Assistentes
Águeda Bernardete Bittencourt	Diretor da Faculdade de Educação
Alcir José Monticelli	Representantes dos Professores Titulares
Angelo Luiz Cortelazzo	Pró-Reitor de Graduação
Antonio Carlos Gilli Martins	Representante dos Professores Assistentes
Adilson Roberto Cardoso	Representante dos Professores Assistentes
Águeda Bernardete Bittencourt	Diretor da Faculdade de Educação
Alcir José Monticelli	Representantes dos Professores Titulares
Angelo Luiz Cortelazzo	Pró-Reitor de Graduação
Antonio Carlos Gilli Martins	Representante dos Professores Assistentes
Adilson Roberto Cardoso	Representante dos Professores Assistentes
Águeda Bernardete Bittencourt	Diretor da Faculdade de Educação
Alcir José Monticelli	Representantes dos Professores Titulares
Angelo Luiz Cortelazzo	Pró-Reitor de Graduação
Antonio Carlos Gilli Martins	Representante dos Professores Assistentes
Adilson Roberto Cardoso	Representante dos Professores Assistentes
Águeda Bernardete Bittencourt	Diretor da Faculdade de Educação
Alcir José Monticelli	Representantes dos Professores Titulares
Angelo Luiz Cortelazzo	Pró-Reitor de Graduação
Antonio Carlos Gilli Martins	Representante dos Professores Assistentes

Suplente

Ana Cristina A. L. de Melio	Representante Comunidade Externa - Prefeitura Municipal
Angela Maria Carneiro Araújo	Representante Geral
Antonia Gomes	Representante dos Servidores Técnicos e Administrativos - Setor Hospitalar
Ana Cristina A. L. de Melio	Representante Comunidade Externa - Prefeitura Municipal
Angela Maria Carneiro Araújo	Representante Geral
Antonia Gomes	Representante dos Servidores Técnicos e Administrativos - Setor Hospitalar

Ana Cristina A. L. de Mello	Representante Comunidade Externa - Prefeitura Municipal
Angela Maria Carneiro Araújo	Representante Geral
Antonia Gomes	Representante dos Servidores Técnicos e Administrativos - Setor Hospitalar
Ana Cristina A. L. de Mello	Representante Comunidade Externa - Prefeitura Municipal
Angela Maria Carneiro Araújo	Representante Geral
Antonia Gomes	Representante dos Servidores Técnicos e Administrativos - Setor Hospitalar
Ana Cristina A. L. de Mello	Representante Comunidade Externa - Prefeitura Municipal
Angela Maria Carneiro Araújo	Representante Geral
Antonia Gomes	Representante dos Servidores Técnicos e Administrativos - Setor Hospitalar

COMISSÃO CENTRAL DE GRADUAÇÃO

Presidente
Angelo Luiz Cortelazzo

Secretário Geral
Maria José Duarte Marcussi Pontes

Membros

Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia
Anselmo Eduardo Diniz	Coordenador do Curso de Engenharia Mecânica
Carlos Roberto Galvão Sobrinho	Coordenador do Curso de História
Celso Dal Ré Carneiro	Coordenador do Curso de Ciências da Terra
Celso Costa Lopes	Coordenador do Curso de Engenharia de Alimentos
Clara G. de Sá Gonçalves Nascimento	Coordenador do Curso de Pedagogia

Suplentes

Ademir José Petenate	Estatística - IMECC
André Franceschi de Angelis	CESET
Beatriz Máscia Daltrini	Engenharia Elétrica - FEEC

Ademir José Petenate	Estatística - IMECC
André Franceschi de Angelis	CESET
Beatriz Máscia Daltrini	Engenharia Elétrica - FEEC
Ademir José Petenate	Estatística - IMECC
André Franceschi de Angelis	CESET
Beatriz Máscia Daltrini	Engenharia Elétrica - FEEC
Ademir José Petenate	Estatística - IMECC
André Franceschi de Angelis	CESET
Beatriz Máscia Daltrini	Engenharia Elétrica - FEEC
Ademir José Petenate	Estatística - IMECC
André Franceschi de Angelis	CESET
Beatriz Máscia Daltrini	Engenharia Elétrica - FEEC
Ademir José Petenate	Estatística - IMECC
André Franceschi de Angelis	CESET
Beatriz Máscia Daltrini	Engenharia Elétrica - FEEC

Convidados

Antonio Faggiani	Diretor Acadêmico - DAC
Elisabete Monteiro de Aguiar Pereira	Assessora da Pró-Reitoria de Graduação - PRG

ATIVIDADES MULTIDISCIPLINARES

A partir da constatação da necessidade de conferir aos profissionais do século XXI uma formação mais abrangente, condizente com a complexidade das situações que este terão que enfrentar, a Unicamp passa, a partir de 1998, a oferecer as disciplinas AM (Atividades Multidisciplinares).

Nestas disciplinas, ser o abordadas temáticas de grande relevância no contexto atual, podendo os alunos escolher livremente aquelas que atendam a seus interesses e curiosidades, enriquecendo sua formação, através do desenvolvimento de seu potencial individual.

É comum, entre os graduandos, o interesse em cursar disciplinas extra-curriculares. Assim, a existência das disciplinas AM vem ampliar as opções de escolha para os alunos.

As disciplinas AM listadas abaixo foram especialmente planejadas pelas Unidades que se dispuseram a contribuir com esta iniciativa. Podem ser cursadas por alunos de qualquer um dos cursos de graduação da Unicamp, sem nenhum pré-requisito:

Unidade Responsável	Disciplina	
FEAGRI	AM001	Ética e Legislação Profissional nas Área de Ciência e Tecnologia
	AM002	Água: Um Objeto Científico
IQ	AM003	Fundamentos de História da Química
FEF	AM004	O Pensar e o Viver o Corpo
FEAGRI	AM001	Ética e Legislação Profissional nas Área de Ciência e Tecnologia
	AM002	Água: Um Objeto Científico
IQ	AM003	Fundamentos de História da Química
FEF	AM004	O Pensar e o Viver o Corpo
FEAGRI	AM001	Ética e Legislação Profissional nas Área de Ciência e Tecnologia
	AM002	Água: Um Objeto Científico
IQ	AM003	Fundamentos de História da Química
FEF	AM004	O Pensar e o Viver o Corpo
FEAGRI	AM001	Ética e Legislação Profissional nas Área de Ciência e Tecnologia
	AM002	Água: Um Objeto Científico
IQ	AM003	Fundamentos de História da Química
FEF	AM004	O Pensar e o Viver o Corpo
FEAGRI	AM001	Ética e Legislação Profissional nas Área de Ciência e Tecnologia
	AM002	Água: Um Objeto Científico
IQ	AM003	Fundamentos de História da Química
FEF	AM004	O Pensar e o Viver o Corpo

CURSOS DE GRADUAÇÃO

Os Cursos de Graduação da Unicamp são apresentados, neste catálogo, por área de conhecimento e, dentro destas, por ordem alfabética do Nome da Unidade responsável pelo oferecimento do curso e, nestas, por ordem alfabética do nome do curso.

A apresentação dos Cursos de Graduação oferecidos pela Unicamp, contém:

- Descrição Sumária da Profissão
- Integralização do Curso
- Limite de Créditos para Matrícula
- Reconhecimento do Curso
- Currículo Pleno, composto por:

a. Núcleo Comum ao Curso
- Disciplinas Obrigatórias

- Disciplinas Eletivas (*)

Observação: Há cursos que possuem também Opção por Língua.

b. Núcleo Específico do Curso/Modalidade
- Disciplinas Obrigatórias

- Disciplinas Eletivas (*)

Observação: Os cursos que possuem uma única modalidade/habilitação apresentam apenas o núcleo comum.

Em cada um dos Núcleos, as disciplinas obrigatórias são apresentadas em ordem alfabética do respectivo código, com as seguintes informações:

- Código das disciplinas (prefixo e número seqüencial)

- Carga horária semanal da disciplina

- Nome da disciplina

Tanto no núcleo comum como no específico, poderão constar um ou mais elencos de disciplinas, com o número de créditos que o aluno deve cumprir. Nesses elencos poderão constar códigos pré-definidos de disciplinas, em ordem alfabética, seguidos da carga horária semanal e nome da disciplina, ou códigos genéricos.

Exemplo:

MA - - - Qualquer disciplina com código do tipo MA - - - (significa qualquer disciplina de prefixo MA)

MA6 - - Qualquer disciplina com código do tipo MA6 - - (significa qualquer disciplina de prefixo MA e primeiro dígito do número seqüencial=6)

- - - - - Qualquer disciplina (significa qualquer disciplina oferecida pela Unicamp e deverá ter autorização expressa da Unidade que a ministra, exceto se a Unidade liberar a autorização).

Os créditos de uma disciplina eletiva serão computados apenas uma vez para efeito de cumprimento do Currículo Pleno, não se permitindo que parte dos créditos da disciplina sejam utilizados em um elenco e o restante em outro elenco.

Na Unicamp, a medida do trabalho escolar dos Cursos de Graduação é o crédito, que corresponde a 15 (quinze) horas de atividade/aula ou atividades acadêmicas supervisionadas (o número de horas semanais é igual ao número de créditos).

- Sugestão oferecida pela Unidade responsável para o cumprimento do Currículo Pleno.

(*) A citação de uma disciplina nas listas de disciplinas aceitáveis para o cumprimento dos créditos eletivos não implica em obrigatoriedade de seu oferecimento regular ou, quando oferecida, na aceitação automática da inscrição dos interessados. As Coordenadorias de Curso responsáveis pelo oferecimento deverão autorizar cada matrícula, dado que, em decorrência da demanda, pode haver necessidade de

ARTES

Artes Cênicas (Diurno)
Dança (Diurno)
Educação Artística - Habilitação Artes Plásticas (Diurno)

Curso de Artes Cênicas

Diretora
Helena Janck

Diretora Associada
Sara Pereira Lopes

Comissão de Graduação

Coordenadora de Curso
Verônica Fabrini M. de Almeida Rocha

Coordenador Associado
Márcio Tadeu Santos Souza

Membros

Laura Argento (Representante Discente Titular)
Luiz Rodrigues Monteiro Junior
Márcio Tadeu Santos Souza
Maria Thaís Lima Santos
Mariana Diana Savioli (Representante Discente Suplente)
Sérgio Ricardo de Carvalho Santos
Verônica Fabrini M. de Almeida Rocha
(Presidente)

Secretária de Graduação

Carmen Ester Cabral Puya

Comissão de Graduação

Endereço para Correspondência
Caixa Postal 6159
Cep:13083-973 - Campinas - São Paulo - Brasil
Tel. (0xx19) 37882439/37882438
Fax (0xx19) 32893140
Email: cgteatro@iar.unicamp.br
slopes@iar.unicamp.br
mima@iar.unicamp.br

26 - Artes Cênicas

Período: Diurno

O Profissional

Bacharel - O Bacharel em Artes Cênicas é o profissional familiarizado com as diferentes linguagens cênicas/teatrais, bem como os diversos sistemas geradores de signos do fenômeno teatral. Possui conhecimento e domínio de técnicas e métodos de trabalho corporal e vocal. Sua bagagem teórica e prática proporciona uma visão do fenômeno teatral como forma de conhecimento crítico da realidade e de uma atuação transformadora e criativa sobre ela. É um profissional preparado para exercer a função de ator, pesquisador e instrutor no campo das Artes Cênicas (teatro, dança, ópera, circo), podendo também dedicar-se, como ator, à televisão e ao cinema. Poderá também atuar como professor universitário.

Exercício Profissional

A Lei Federal nº 006533, de 24/05/1978, regulamenta o exercício da profissão.
O Decreto Federal nº 082385, de 05/10/1978, regulamenta a mencionada lei.
O Decreto Federal nº 095971, de 27/04/1988, altera o decreto acima.

Integralização

Para graduar-se neste curso, o aluno deverá perfazer o total de 184 créditos, equivalentes a 2760 horas. O curso poderá ser integralizado em 08 semestres, conforme sugestão da unidade para o cumprimento do currículo pleno, sendo o prazo máximo de integralização 12 semestres.

Reconhecimento

Reconhecido pela Portaria MEC nº 000961, de 24/06/1992.

Limite de Créditos para matrícula semestral - Máximo de 40 créditos.

CURRICULO PLENO

Núcleo Comum ao Curso:

AC105	02	Canto para o Ator I	AC109	02	Música e Ritmo I
AC113	02	Formas Espetaculares no Ocidente	AC119	04	Filosofia e História da Arte: Filosofia da Arte

UNICAMP - CATÁLOGO DOS CURSOS DE PÓS-GRADUAÇÃO - 2001

AC128	02	História do Teatro: Formas Espetaculares no Brasil I	AC105	02	Canto para o Ator I
AC105	02	Canto para o Ator I	AC113	02	Formas Espetaculares no Ocidente
AC113	02	Formas Espetaculares no Ocidente	AC113	02	Formas Espetaculares no Ocidente
AC105	02	Canto para o Ator I	AC105	02	Canto para o Ator I
AC109	02	Música e Ritmo I	AC109	02	Música e Ritmo I
AC113	02	Formas Espetaculares no Ocidente	AC113	02	Formas Espetaculares no Ocidente
AC113	02	Formas Espetaculares no Ocidente	AC109	02	Música e Ritmo I
AC105	02	Canto para o Ator I	AC105	02	Canto para o Ator I
AC105	02	Canto para o Ator I	AC113	02	Formas Espetaculares no Ocidente
AC113	02	Formas Espetaculares no Ocidente	AC105	02	Canto para o Ator I
AC105	02	Canto para o Ator I	AC109	02	Música e Ritmo I
AC109	02	Música e Ritmo I	AC113	02	Formas Espetaculares no Ocidente
AC109	02	Música e Ritmo I	AC105	02	Canto para o Ator I
AC105	02	Canto para o Ator I	AC105	02	Canto para o Ator I
AC113	02	Formas Espetaculares no Ocidente	AC113	02	Formas Espetaculares no Ocidente
AC105	02	Canto para o Ator I	AC109	02	Música e Ritmo I
AC113	02	Formas Espetaculares no Ocidente	AC109	02	Música e Ritmo I
AC105	02	Canto para o Ator I			

Sugestão oferecida pela unidade responsável para o cumprimento do currículo pleno:

01º Semestre: 31 Créditos				**02º Semestre:** 33 Créditos			
AC111(02)	AC112(02)	AC113(04)	AC114(04)	AC121(02)	AC122(02)	AC123(04)	AC124(04)
AC115(04)	AC111(02)	AC112(02)	AC113(04)	AC121(02)	AC122(02)	AC123(04)	AC124(04)
AC114(04)	AC115(04)	AC115(04)		AC121(02)	AC122(02)	AC123(04)	AC124(04)
03º Semestre: 20 Créditos				**04º Semestre:** 20 Créditos			
AC131(02)	AC132(02)	AC133(04)	AC134(04)	AC141(02)	AC142(02)	AC143(04)	AC144(04)
AC135(04)	AC136(04)			AC143(04)	AC144(04)		
05º Semestre: 31 Créditos				**06º Semestre:** 31 Créditos			
AC111(02)	AC112(02)	AC113(04)	AC114(04)	AC111(02)	AC112(02)	AC113(04)	AC114(04)
AC115(04)	AC114(04)	AC115(04)		AC115(04)	AC113(04)	AC114(04)	AC115(04)
07º Semestre: 31 Créditos				**08º Semestre:** 31 Créditos			
AC111(02)	AC112(02)	AC113(04)	AC114(04)	AC111(02)	AC112(02)	AC113(04)	AC114(04)
AC115(04)							

Curso de Dança

Diretora
Helena Janck

Diretora Associada
Sara Pereira Lopes

Comissão de Graduação

Coordenadora de Curso
Verônica Fabrini M. de Almeida Rocha

Coordenador Associado
Márcio Tadeu Santos Souza

Membros

Laura Argento (Representante Discente Titular)
Luiz Rodrigues Monteiro Junior
Márcio Tadeu Santos Souza
Maria Thais Lima Santos
Mariana Diana Savioli (Representante Discente Suplente)
Sérgio Ricardo de Carvalho Santos
Verônica Fabrini M. de Almeida Rocha
(Presidente)

Secretária de Graduação

Carmen Ester Cabral Puya

Comissão de Graduação

Endereço para Correspondência
Caixa Postal 6159
Cep:13083-973 - Campinas - São Paulo - Brasil
Tel. (0xx19) 37882439/37882438
Fax (0xx19) 32893140
Email: cgdança@iar.unicamp.br
graziela@iar.unicamp.br
mima@iar.unicamp.br

CORPO DOCENTE

Instituto de Artes

Departamento de Artes Cênicas

Marcio Aurelio Pires de Almeida, MS-3, RDIDP
Renato Cohen, MS-3, RTC
Sara Pereira Lopes, MS-3, RDIDP
Maria Thais Lima Santos, MS-2, RDIDP
Sergio Ricardo de Carvalho Santos, D, RDIDP
Maria Lucia Levy Candeias, MS-3, RDIDP
Rubens Jose Souza Brito, MS-3, RDIDP
Veronica Fabrini Machado de Rocha, MS-3, RDIDP
Marcio Tadeu Souza Santos, G, RDIDP
Luiz Rodrigues Monteiro Junior, A, RTC

Departamento de Artes Corporais

Marcio Aurelio Pires de Almeida, MS-3, RDIDP
Renato Cohen, MS-3, RTC
Sara Pereira Lopes, MS-3, RDIDP
Maria Thais Lima Santos, MS-2, RDIDP
Sergio Ricardo de Carvalho Santos, D, RDIDP
Maria Thais Lima Santos, MS-2, RDIDP
Sergio Ricardo de Carvalho Santos, D, RDIDP
Marcio Aurelio Pires de Almeida, MS-3, RDIDP
Renato Cohen, MS-3, RTC
Maria Lucia Levy Candeias, MS-3, RDIDP
Rubens Jose Souza Brito, MS-3, RDIDP
Veronica Fabrini Machado de Rocha, MS-3, RDIDP
Marcio Tadeu Souza Santos, G, RDIDP
Luiz Rodrigues Monteiro Junior, A, RTC
Maria Lucia Levy Candeias, MS-3, RDIDP
Rubens Jose Souza Brito, MS-3, RDIDP

Departamento de Artes Plásticas

Marcio Aurelio Pires de Almeida, MS-3, RDIDP
Renato Cohen, MS-3, RTC
Sara Pereira Lopes, MS-3, RDIDP
Maria Thais Lima Santos, MS-2, RDIDP
Sergio Ricardo de Carvalho Santos, D, RDIDP
Maria Lucia Levy Candeias, MS-3, RDIDP
Rubens Jose Souza Brito, MS-3, RDIDP
Veronica Fabrini Machado de Rocha, MS-3, RDIDP
Marcio Tadeu Souza Santos, G, RDIDP
Luiz Rodrigues Monteiro Junior, A, RTC

DISCIPLINAS DOS CURSOS DE GRADUAÇÃO

INFORMAÇÕES GERAIS

As disciplinas oferecidas pelas unidades encontram-se identificadas a seguir. As informações são, na ordem em que aparecem, as seguintes:

• Código da Disciplina

• Nome da Disciplina

• Conjunto de letras e números, significando:

- OF Período de ofererimento da disciplina, de acordo com a convenção:
- S-1 - 1º período letivo
- S-2 - 2º período letivo
- S-5 - Ambos os períodos letivos. Só terá direito à matrícula o aluno de curso que, pela sugestão para o cumprimento do currículo, apresente a disciplina no semestre correspondente.
- S-6 - A Critério da Unidade de Ensino.
- T Horas-aula semanais de teoria.
- P Horas-aula semanais de prática.
- L Horas-aula semanais de laboratório.
- HS Horas-aula semanais.
- SL Horas-aula semanais em sala.
- C Créditos da disciplina, relativos a um período letivo de quinze semanas.

• Pré-Requisito

É a disciplina ou disciplinas nas quais o aluno deve obter aproveitamento necessário para a matrícula em outra disciplina, desde que considerado indispensável do ponto de vista acadêmico

• Pré-Requisito Pleno

É a disciplina ou disciplinas nas quais o aluno deve obter aprovação, para matrícula em outra disciplina.

• Pré-Requisito Parcial

É a disciplina ou disciplinas nas quais o aluno deve obter a freqüência mínima estabelecida pelo Departamento e média final maior ou igual a três (3,0), para matricular-se em outra disciplina. São identificadas nos pré-requisitos com um asterisco (*) na frente do código da disciplina.

• Pré-Requisito Especiais:

AA200 - Autorização da Coordenadoria que oferece a disciplina.

AA4nn - O aluno deve possuir CP (Coeficiente de Progressão) maior ou igual a 0,nn.

EX: AA475 - significa que o aluno, para cursar esta disciplina, deve ter cursado pelo menos 75% do curso (CP - Coeficiente de Progressão) maior ou igual a 0,75.

Observação:
Os códigos das disciplinas nos pré-requisitos podem estar separados por "espaço" ou /, de acordo com a convenção:

/ (barra) - significa "**ou**"
Espaço - significa "**e**"

• Ementa

A ementa descreve sucintamente o assunto relacionado com a disciplina. Em algumas disciplinas, principalmente naquelas relacionadas com Tópicos Especiais, as ementas serão oferecidas pelas Unidades de Ensino correspondentes, na época da oferta dessas disciplinas

DISCIPLINAS

AC104 Fundamentos da Expressão e Comunicação Humanas: Introdução ao Teatro I
OF:S-1 T:02 P:00 L:00 HS:02 SL:02 C:02
Ementa: Desenvolvimento no aluno de um ponto de vista próprio sobre a realidade teatral, através do contato com os profissionais atuantes na área.
AC105 Canto para o Ator I
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Pré-Req.: AC611
Ementa: Desenvolvimento das potencialidades musicais do aluno através do canto individual e do canto coral, como elemento de qualificação para o trabalho do ator.
AC108 História do Teatro: Formas Espetaculares no Brasil I
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Ementa: Proporciona ao aluno o conhecimento das principais configurações do espetáculo teatral no Brasil. Estudo das origens do fenômeno teatral no Brasil. Manifestações teatrais de aculturação ibérica: autos e entremezes. Aencenação jesuítica e sua organização segundo as relações espaciais palco-platéia: o palco elisabetano, o palco italiano e o palco sem-limite. Folguedos populares de aculturação indígena, africana e ibérica.
AD101 Consciência Corporal I
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Ementa: Desenvolver o conceito de corpo como uma unidade psico-física. Noção de eixo e da integração das partes do corpo em relação a esse eixo. (Articulações). Trabalha equilíbrio, fluência e a flexibilidade. Estimula, através da improvisação e da criatividade, a incorporação destes conceitos às técnicas de dança.
AD204 Exercícios Técnicos de Dança II
OF:S-2 T:00 P:06 L:00 HS:06 SL:06 C:06
Pré-Req.: AD104 / *AA200
Ementa: Combinação de alongamentos laterais nos movimentos pendulares em flexões no plano vertical. Desenvolvimento de tônus muscular através de variações dinâmicas com os elementos básicos da dança.
AD434 Elementos de linguagem musical para dançarinos II
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Pré-Req.: AD204 / AD334
Ementa: Depois do conhecimento básico da linguagem musical oferecido nas disciplinas AD-109 e AD334, o aluno terá, neste curso, condições para poder pensar na relação entre dança e música, penetrando no universo de concepções da composição coreográfica em relação às estruturas musicais. Este estudo se dá tanto através de análise de coreografias em vídeo, quanto de trabalhos práticos de dança orientados.
AM002 Água: Um Objeto Científico
OF:S-6 T:02 P:00 L:00 HS:02 SL:02 C:02
Ementa: A água e a fonte do saber. As propriedades físicas e químicas da água. Hidrologia e ciclo da água. A bacia hidrográfica como uma unidade hidrológica. A água e a vida. A dinâmica das paisagens fluviais. A economia da água: disponibilidades, demandas e conflitos. Água: um bem econômico. Qualidade da água. Erosão hídrica. Água e saúde pública. A ocorrência, intensificação e controle de inundação. A gestão e planejamento dos recursos hídricos.
AM020 Grandes Temas da Atualidade
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Pré-Req.: AA200
Ementa: Ciclo de palestras abordando grandes temas da atualidade desenvolvido por especialistas ligados aos mesmos.
AU001 Projeto de Graduação I
OF:S-1 T:00 P:12 L:00 HS:12 SL:12 C:12
Pré-Req.: AP120
Ementa: Projeto arquitetônico e de desenho urbano complexo com abordagem interdisciplinar. Sequência completa das etapas de projeto profissional: estudos preliminares, análise de programa funcional, definição de partido, anteprojeto, maquetes de estudo, plano de massa para implantação na malha urbana.
AU815 Tecnologia V: Gerenciamento
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Pré-Req.: AA475
Ementa: Sistemas de gerenciamento e planejamento de empreendimentos. Fases da construção civil. Estruturas organizacionais para gerenciamento das operações. Controle de custos e de prazos. Análise de viabilidade técnica, econômica e financeira de empreendimentos. A inflação e seus efeitos no planejamento. Orçamentação com uso de softwares.
AP109 Apreciação Musical I
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Ementa: Noções da estrutura da música ocidental em seus aspectos melódico, rítmico, harmônico e timbrístico, visando a uma escuta consciente, crítica e fluente.
AP202 Estética e História da Arte II
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Pré-Req.: AP102
Ementa: Manifestações artísticas da Pré-história e dos povos da antigüidade até o Império Romano. Arquitetura e arte da Idade Média: o pensamento medieval e as formas artísticas, as transformações artísticas no século XIV.
BA110 Anatomia Humana I
OF:S-1 T:00 P:02 L:00 HS:04 SL:04 C:04
Ementa: Conceitos sobre a construção geral do corpo humano. Aparelho Locomotor: Sistema ósseo, articular e muscular. Sistema circulatório. Sistema linfático e Sistema respiratório.
BA500 Iniciação Científica em Anatomia I
OF:S-1 T:00 P:02 L:00 HS:02 SL:02 C:02
Pré-Req.: AA200 / AA430

Capítulo 6

Considerações Finais

Este trabalho teve como objetivo apresentar o projeto de uma aplicação típica de utilização da meta-linguagem XML e de alguns padrões relacionados, na troca de informação entre diferentes formatos para o catálogo de cursos de graduação e pós-graduação da Universidade e sua disponibilização nos formatos HTML, RTF, PS e PDF. O modelo pretende sanar importantes deficiências na atual metodologia de geração desses catálogos.

A principal contribuição deste trabalho consistiu em definir um padrão de documentos XML para a estruturação das informações, permitindo a automatização da sua geração através de formulários Web; resolveu-se o problema existente de interoperabilidade na troca de informação entre os coordenadores dos cursos e a administração da Universidade, assim como a substituição do formato interno "ad hoc" de representação da informação pela especificação em XML com seu respectivo DTD, e a eliminação de redundância nas informações disponibilizadas em diferentes partes dos documentos originais, levando à geração e publicação mais rápida e eficiente dos catálogos de graduação e pós-graduação.

6.1 Trabalhos Futuros

Um próximo trabalho consistiria na utilização da tecnologia XML para solucionar problemas relacionados ao armazenamento, atualização e recuperação de informação no formato XML. Sistemas de banco de dados relacionais oferecem confiabilidade, escalabilidade e desempenho, além de ferramentas para recuperação/backup de dados, sendo uma boa opção para resolver esses problemas. Será necessário a definição de métodos de mapeamento de documentos XML para banco de dados relacionais e vice-versa, como também métodos de transformação de consultas em linguagens de consultas XML (como por exemplo, XQuery 1.0: An XML Query Language [4]) para consultas SQL (Structured Query Language). Desta forma um documento XML pode ser mapeado em tabelas seguindo um esquema relacional e consultas formuladas em uma linguagem de consulta XML transformadas em consultas SQL.

Referências Bibliográficas

[1] Bray, T., Hollander, D., Layman, A. *Namespaces in XML* World Wide Web Consortium (W3C) Recommendation, 1999, http://www.w3.org/TR/REC-xml-names/.

[2] Adler, S., Berglund, A., Caruso, J., Deach, S. *Extensible Stylesheet Language (XSL) Version 1.0.* World Wide Web Consortium (W3C) Recommendation, 2001, http://www.w3.org/TR/xsl/.

[3] Bakken, S., Aulbach, A., Schmid, E., Winstead, J., Wilson, L., Lerdorf, R., Zmievski, A., Ahto, J. *Manual do PHP.* 2002, http://www.php.net/manual/pt_BR/.

[4] Boag, S., Chamberlin, D., Fernandez, M. F., Florescu, D., Robie, J., Siméon, J. *XQuery 1.0: An XML Query Language.* World Wide Web Consortium (W3C), 2002, http://www.w3.org/TR/xquery/.

[5] Bos, B., Lie, H. W., Lilley, C., Jacobs, I. *Cascading Style Sheets, level 2 (CSS2) Specification.* World Wide Web Consortium (W3C) Recommendation, 1998, http://www.w3.org/TR/1998/REC-CSS2-19980512.

[6] Bourret, R. *Declaring Elements and Attributes in an XML DTD.* 1999, http://www.rpbourret.com/xml/xmldtd.htm.

[7] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. *Extensible Markup Language (XML) 1.0 (Second Edition).* World Wide Web Consortium (W3C) Recommendation, 2000, http://www.w3.org/TR/2000/REC-xml-20001006.

[8] Brickley, D., Swick, R. R. *Resource Description Framework* World Wide Web Consortium (W3C) Recommendation, 2000, http://www.w3.org/RDF/.

[9] Cabaniss, J. H. *EBNF for XML 1.0.* http://www.jelks.nu/XML/xmlebnf.html. (visitado em Maio de 2002).

[10] Carey, M., Kiernan, J., Shanmugasundaram, J., Shekita, E., Subramanian, S. *XPERANTO: A Middleware for Publishing Object-Relational Data as XML Documents.* International Conference on Very Large Data Bases (VLDB), Cairo, Egito, 2000.

[11] Clark., J.*Jade - James' DSSSL Engine.* http://www.jclark.com/jade/. (visitado em Setembro de 2001).

[12] Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D. *XML-QL: A Query Language for XML.* World Wide Web Consortium (W3C), 1998, http://www.w3.org/TR/NOTE-xml-ql/.

[13] *DocBook Open Repository.* SourceForge, http://docbook.sourceforge.net/. (visitado em Fevereiro de 2001).

[14] *Document Object Model (DOM).* World Wide Web Consortium (W3C) Recommendation, 2002, http://www.w3.org/DOM/.

[15] Fallside, D. C. *XML Schema* World Wide Web Consortium (W3C) Recommendation, 2001, http://www.w3.org/XML/Schema/.

[16] Fernandéz, M., Tan, W., Suciu, D. *SilkRoute: Trading between Relations and XML.* WWW9/Computer Networks, 2000.

[17] Garshol, L. M. *BNF and EBNF: What are they and how do they work?.* MSc student of University of Oslo, http://www.garshol.priv.no/download/text/bnf.html. (visitado em Maio de 2002).

[18] Goossens, M., Saarela, J. *A Practical Introduction to SGML.* CERN, CN Division, Switzerland, 1995, http://www.irb.hr:80/ cern/WWW/publications/sgmlen/sgmlen.html.

[19] *Guide to the W3C XML Specification ("XMLspec") DTD, Version 2.1.* World Wide Web Consortium (W3C) Recommendation, 1998, http://www.w3.org/XML/1998/06/xmlspec-report.

[20] Gundavaram, S. *CGI Programming on the World Wide Web.* O'Reilly & Associates, Inc., 1996, http://www.oreilly.com/openbook/cgi.

[21] Harold, E. *Chapter 18 of the XML Bible, Second Edition: XSL Formatting Objects.* 2002, http://www.ibiblio.org/xml/books/bible2/chapters/ch18.html.

[22] ISO/IEC 10179:1996 *DSSSL - Document Style Semantics and Specification Language.* 1996, http://www.oasis-open.org/cover/dsssl.html.

[23] Kay, M. *Saxon XSLT Processor* SouceForge.net, 2001, http://saxon.sourceforge.net/.

[24] Laurent, S. *Why XML?.* 1998, http://www.simonstl.com/articles/whyxml.htm.

[25] Megginson, D. *SAX 2.0: The Simple API for XML.* Megginson Technologies Ltd. 2000, http://www.megginson.com/SAX/index.html.

[26] Møller, A., Schwartzbach, M. I. *The XML Revolution, Technologies for the future Web.* BRICS, University of Aarhus, 2002, http://www.brics.dk/ amoeller/XML/.

[27] Pemberton, S. *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition).* World Wide Web Consortium (W3C) Recommendation, 2000, http://www.w3.org/TR/2002/REC-xhtml1-20020801.

[28] Raggett, D., Le Hors, A., Jacobs, I. *HTML 4.01 Specification.* World Wide Web Consortium (W3C) Recommendation, 1999, http://www.w3.org/TR/html4.

[29] Refsnes, J. E. *Introduction to DTD.* XMLFiles.com, http://xmlfiles.com/dtd/dtd_intro.asp. (visitado em Janeiro de 2001).

[30] Robie, J., Lapp, J., Schach, D. *XML Query Language (XQL).* World Wide Web Consortium (W3C), 1998, http://www.w3.org/TandS/QL/QL98/pp/xql.html.

[31] Rossum, G. *Python Tutorial.* PythonLabs, 2002, http://www.python.org/doc/current/tut/tut.html.

[32] *Unicorn Formatting Objects.* Unicorn Enterprises SA, http://www.unicorn-enterprises.com/products_ufo.html. (visitado em Junho de 2002).

[33] Wall, L., Christiansen, T., Orwant, J. *Programming Perl (Third Edition)).* O'Reilly & Associates, Inc., 2000, http://octopus.cdut.edu.cn/ yf17/perlcd/prog/.

[34] Walsh, N. *A Technical Introduction to XML.* 1998, http://www.xml.com/pub/a/98/10/guide0.html.

[35] Walsh, N., Muellner, L. *DocBook: The Definitive Guide.* O'Reilly & Associates, Inc., 1999, http://www.oreilly.com/catalog/docbook/.

Apêndice A

Documentos Referentes ao Catálogo da Pós-Graduação

A seguir será mostrada parte dos documentos utilizados para a geração do Catálogo de Pós-Graduação do IC.

A.1 Documento XML para o Catálogo da Pós-Graduação

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE catalogo-pos SYSTEM "DtdCatalogoPos.dtd">

<catalogo-pos>

<capa>
<cabecalho>
<paragrafo>GOVERNO DO ESTADO DE SÃO PAULO</paragrafo>
<paragrafo>UNIVERSIDADE ESTADUAL DE CAMPINAS</paragrafo>
</cabecalho>
<titulo>INSTITUTO DE COMPUTAÇÃO</titulo>
<subtitulo>CATÁLOGO DOS CURSOS DE PÓS-GRADUAÇÃO</subtitulo>
<ano>2001</ano>
</capa>

<apresentacao>
<titulo>INSTITUTO DE COMPUTAÇÃO</titulo>
<diretor><nome>Tomasz Kowaltowski</nome></diretor>

<diretor-associado><nome>Ricardo de Oliveira Anido</nome></diretor-associado>
<secretaria><nome>Maria Magali Chaguri</nome></secretaria>
</apresentacao>

<corpo>

<departamento>
<nome-departamento>DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO
</nome-departamento>
<membro ms="6"><nome>Tomasz Kowaltowski</nome><r>RDIDP</r></membro>
<membro ms="5"><nome>Célio Cardoso Guimarães</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Edmundo Roberto Mauro Madeira</nome><r>RDIDP</r>
</membro>
<membro ms="4"><nome>Nelson Luís Saldanha da Fonseca</nome><r>RDIDP</r>
</membro>
<membro ms="4"><nome>Paulo Lício de Geus</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Ricardo de Oliveira Anido</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Rogério Drummond Burnier Pessoa de Mello Filho</nome>
<r>RDIDP</r></membro>
<membro ms="3"><nome>Alexandre Xavier Falcão</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Arthur João Catto</nome><r>RTC</r></membro>
<membro ms="3"><nome>Guido Costa Souza de Araújo</nome><r>RDIDP</r>
</membro>
<membro ms="3"><nome>Maria Beatriz Felgar de Toledo</nome><r>RDIDP</r>
</membro>
<membro ms="3"><nome>Mario Lúcio Côrtes</nome><r>RTC</r></membro>
<membro ms="3"><nome>Paulo César Centoducatte</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Ricardo Pannain</nome><r>RTP</r></membro>
<membro ms="2"><nome>Fernando Antonio Vanini</nome><r>RTP</r></membro>
<membro ms="2"><nome>Silvia Helena Machado de Oliveira</nome><r>RDIDP</r>
</membro>
</departamento>

<departamento><nome-departamento>DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO
</nome-departamento>
<membro ms="5"><nome>Claudia Maria Bauzer Medeiros</nome><r>RDIDP</r>
</membro>
<membro ms="4"><nome>Ariadne Maria Brito Rizzoni Carvalho</nome><r>RDIDP</r>
</membro>
<membro ms="4"><nome>Geovane Cayres Magalhães</nome><r>RTP</r></membro>
<membro ms="4"><nome>Hans Kurt Edmund Liesenberg</nome><r>RDIDP</r>
</membro>
<membro ms="4"><nome>Heloisa Vieira da Rocha</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Jacques Wainer</nome><r>RDIDP</r></membro>

<membro ms="4"><nome>Luiz Eduardo Buzato</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Maria Cecília Calani Baranauskas</nome><r>RDIDP</r>
</membro>
<membro ms="4"><nome>Neucimar Jerônimo Leite</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Cecília Mary Fischer Rubira</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Eliane Martins</nome><r>RDIDP</r></membro>
<membro ms="2"><nome>Thelma Cecília dos Santos Chiossi</nome><r>RTC</r>
</membro>
<membro ms="1"><nome>Sindo Vasquez Dias</nome><r>RTP</r></membro>
</departamento>

<departamento><nome-departamento>DEPARTAMENTO DE TEORIA DA COMPUTAÇÃO
</nome-departamento>
<membro ms="6"><nome>Cláudio Leonardo Lucchesi</nome><r>RDIDP</r></membro>
<membro ms="5"><nome>Jorge Stolfi</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Arnaldo Vieira Moura</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Célia Picinin de Mello</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Cid Carvalho de Souza</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>João Carlos Setubal</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>João Meidanis</nome><r>RDIDP</r></membro>
<membro ms="4"><nome>Pedro Jussieu de Rezende</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Anamaria Gomide</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Flávio Keidi Miyazawa</nome><r>RDIDP</r></membro>
<membro ms="3"><nome>Ricardo Dahab</nome><r>RDIDP</r></membro>
</departamento>

<comissao-pos>
<membro-comissao classificacao="Coordenador"><nome>Pedro Jussieu de Rezende</nome>
</membro-comissao>
<membro-comissao classificacao="Membro"><nome>Luiz Eduardo Buzato</nome>
</membro-comissao>
<membro-comissao classificacao="Membro"><nome>Maria Cecília Baranauskas</nome>
</membro-comissao>
<membro-comissao classificacao="Suplente"><nome>Adilson Luiz Bonifácio</nome>
</membro-comissao>
<membro-comissao classificacao="Suplente"><nome>Flávio Keidi Miyazawa</nome>
</membro-comissao>
<membro-comissao classificacao="Representante"><nome>Sandro Rigo</nome>
</membro-comissao>
</comissao-pos>

<introducao>
<paragrafo-tabulacao>

O Programa de Pós-graduação em Computação, teve início em março de 1977, no Instituto de Matemática, Estatística e Computação Científica (IMECC) e passou a funcionar junto ao Instituto de Computação a partir da criação deste pela Deliberação Consu-A-1, de 26 de março de 1996.

</paragrafo-tabulacao>
<paragrafo-tabulacao>
Estão em funcionamento os Cursos de Mestrado e Doutorado em Computação.
</paragrafo-tabulacao>
<paragrafo>
Várias informações atualizadas estão disponíveis na web: http://www.dcc.unicamp.br/cpg
</paragrafo>
</introducao>

<corpo-docente>
<professor><nome>Alexandre Xavier Falcão</nome><graduacao><curso>Eng. Elétr.
</curso>
<universidade>UFPe</universidade><local>PE</local><ano>1988</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1993</ano></mestrado>
<doutorado><universidade>Unicamp</universidade><ano>1996</ano></doutorado>
</professor>

<professor><nome>Anamaria Gomide</nome><graduacao><curso>Bach. Matemática
</curso>
<universidade>UnB</universidade><ano>1974</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1999</ano></mestrado>
</professor>

<professor><nome>Ariadne Maria Brito Rizzoni Carvalho</nome><graduacao>
<curso>Anal. Sistemas</curso>
<universidade>PUCCAMP</universidade><ano>1979</ano></graduacao>
<doutorado><universidade>Reading</universidade><ano>1989</ano></doutorado>
<livre-docente></livre-docente>
</professor>

<professor><nome>Arnaldo Vieira Moura</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>ITA</universidade><ano>1973</ano></graduacao>
<mestrado><universidade>ITA</universidade><ano>1976</ano></mestrado>
<doutorado><universidade>Univ. of California</universidade><local>Berkeley</local>
<ano>1980</ano></doutorado>
<livre-docente></livre-docente>
</professor>

<professor><nome>Arthur João Catto</nome><graduacao><curso>Eng. Civil</curso>
<universidade>USP</universidade><local>São Carlos</local><ano>1970</ano>
</graduacao>

<mestrado><universidade>USP</universidade><local>São Carlos</local><ano>1977</ano
</mestrado>
<mestrado><universidade>Manchester Univ.</universidade><ano>1978</ano></mestrado>
<doutorado><universidade>Manchester Univ.</universidade><ano>1981</ano></doutorad
</professor>

<professor><nome>Cecília Mary Fischer Rubira</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1985</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1990</ano></mestrado>
<doutorado><universidade>Univ. of Newcastle upon Tyne</universidade><ano>1994</ano>
</doutorado>
</professor>

<professor><nome>Célia Picinin de Mello</nome><graduacao><curso>Lic. Mat.</curso>
<universidade>FFCL</universidade><local>Rio Claro</local><ano>1974</ano>
</graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1984</ano></mestrado>
<doutorado><universidade>UFRJ</universidade><ano>1992</ano></doutorado>
</professor>

<professor><nome>Célio Cardoso Guimarães</nome><graduacao>
<curso>Eng. Elétr.</curso>
<universidade>ITA</universidade><ano>1965</ano></graduacao>
<mestrado><universidade>Case Western Reserve Univ.</universidade>
<local>Cleveland</local><ano>1970</ano></mestrado>
<doutorado><universidade>Case Western Reserve Univ.</universidade>
<local>Cleveland</local><ano>1973</ano></doutorado>
</professor>

<professor><nome>Cid Carvalho de Souza</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>PUC</universidade><local>RJ</local><ano>1985</ano></graduacao>
<mestrado><universidade>PUC</universidade><local>RJ</local><ano>1989</ano>
</mestrado>
<doutorado><universidade>Univ. Catholique de Louvain</universidade>
<local>Bélgica</local><ano>1993</ano></doutorado>
<livre-docente></livre-docente>
</professor>

<professor><nome>Claudia Maria Bauzer Medeiros</nome><graduacao>
<curso>Eng. Elétr.</curso>
<universidade>PUC</universidade><local>RJ</local><ano>1976</ano></graduacao>
<mestrado><universidade>PUC</universidade><local>RJ</local><ano>1979</ano>
</mestrado>
<doutorado><universidade>Univ. Waterloo</universidade><local>Canadá</local>

<ano>1985</ano></doutorado>
<livre-docente><universidade>Unicamp</universidade><ano>1992</ano></livre-docente>
</professor>

<professor><nome>Cláudio Leonardo Lucchesi</nome><graduacao>
<curso>Eng. Elétr.</curso>
<universidade>USP</universidade><ano>1969</ano></graduacao>
<mestrado><universidade>Univ. Waterloo</universidade><local>Canadá</local>
<ano>1970</ano></mestrado>
<doutorado><universidade>Univ. Waterloo</universidade><local>Canadá</local>
<ano>1976</ano></doutorado>
</professor>

<professor><nome>Edmundo Roberto Mauro Madeira</nome><graduacao>
<curso>Eng. Civil</curso>
<universidade>Unicamp</universidade><ano>1980</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1985</ano></mestrado>
<doutorado><universidade>Unicamp</universidade><ano>1991</ano></doutorado>
</professor>

<professor><nome>Eliane Martins</nome><graduacao><curso>Bach. Inf.</curso>
<universidade>UFRJ</universidade><ano>1977</ano></graduacao>
<mestrado><universidade>COPPE</universidade><local>UFRJ</local>
<ano>1993</ano></mestrado>
<doutorado><universidade>École Nationale Supérieure de L'Aéronautique et de L'Espace
</universidade><local>Toulouse</local><ano>1992</ano></doutorado>
</professor>

<professor><nome>Flávio Keidi Miyazawa</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>UFMS</universidade><ano>1990</ano></graduacao>
<mestrado><universidade>USP</universidade><ano>1993</ano></mestrado>
<doutorado><universidade>USP</universidade><ano>1997</ano></doutorado>
</professor>

<professor><nome>Geovane Cayres Magalhães</nome><graduacao>
<curso>Eng. Civil</curso>
<universidade>UFBa</universidade><ano>1971</ano></graduacao>
<mestrado><universidade>PUC</universidade><local>RJ</local><ano>1976</ano>
</mestrado>
<doutorado><universidade>Univ. de Toronto</universidade><ano>1981</ano></doutorado>
<livre-docente></livre-docente>
</professor>

<professor><nome>Guido Costa Souza de Araújo</nome><graduacao>
<curso>Eng. Elétr.</curso>

<universidade>UFPe</universidade><ano>1985</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1991</ano></mestrado>
<mestrado><universidade>Princeton Univ.</universidade><ano>1994</ano></mestrado>
<doutorado><universidade>Princeton Univ.</universidade><ano>1997</ano></doutorado>
</professor>

<professor><nome>Hans Kurt Edmund Liesenberg</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1976</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1980</ano></mestrado>
<doutorado><universidade>Univ. Newcastle Upon Tyne</universidade><ano>1985</ano>
</doutorado>
</professor>

<professor><nome>Heloisa Vieira da Rocha</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1975</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1981</ano></mestrado>
<doutorado><universidade>Unicamp</universidade><ano>1991</ano></doutorado>
</professor>

<professor><nome>Jacques Wainer</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>USP</universidade><ano>1982</ano></graduacao>
<doutorado><universidade>Pensylvania State Univ.</universidade><ano>1991</ano>
</doutorado><livre-docente></livre-docente>
</professor>

<professor><nome>João Carlos Setubal</nome><graduacao><curso>Eng. Mec.</curso>
<universidade>USP</universidade><ano>1979</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1987</ano></mestrado>
<doutorado><universidade>Univ. Washington</universidade><ano>1992</ano>
</doutorado><livre-docente></livre-docente>
</professor>

<professor><nome>João Meidanis</nome><graduacao><curso>Bach. Mat.</curso>
<universidade>USP</universidade><ano>1980</ano></graduacao>
<mestrado><universidade>USP</universidade><ano>1984</ano></mestrado>
<mestrado><universidade>Univ. Wisconsin</universidade><ano>1989</ano></mestrado>
<doutorado><universidade>Univ. Wisconsin</universidade><ano>1992</ano></doutorado>
<livre-docente><universidade>Unicamp</universidade><ano>1996</ano></livre-docente>
</professor>

<professor><nome>Jorge Stolfi</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>USP</universidade><ano>1973</ano></graduacao>
<mestrado><universidade>USP</universidade><ano>1979</ano></mestrado>
<doutorado><universidade>Stanford Univ.</universidade><ano>1989</ano></doutorado>

<livre-docente><universidade>Unicamp</universidade><ano>1996</ano></livre-docente>
</professor>
<professor><nome>Luiz Eduardo Buzato</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1985</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1990</ano></mestrado>
<doutorado><universidade>Univ. of New Castle Upon Tyne</universidade>
<local>UK</local><ano>1994</ano></doutorado>
</professor>
<professor><nome>Maria Beatriz Felgar de Toledo</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1980</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1986</ano></mestrado>
<doutorado><universidade>Univ. Lancaster</universidade><ano>1992</ano></doutorado>
</professor>
<professor><nome>Maria Cecília Calani Baranauskas</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1976</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1981</ano></mestrado>
<doutorado><universidade>Unicamp</universidade><ano>1993</ano></doutorado>
</professor>
<professor><nome>Mário Lúcio Côrtes</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>ITA</universidade><ano>1973</ano></graduacao>
<mestrado><universidade>USP</universidade><ano>1980</ano></mestrado>
<doutorado><universidade>Stanford</universidade><ano>1987</ano></doutorado>
<livre-docente></livre-docente>
</professor>
<professor><nome>Nelson Luís Saldanha da Fonseca</nome><graduacao>
<curso>Eng. Elétr.</curso>
<universidade>PUC</universidade><local>RJ</local><ano>1984</ano></graduacao>
<mestrado><universidade>Univ. Southern California</universidade><ano>1993</ano>
</mestrado>
<doutorado><universidade>Univ. Southern California</universidade><ano>1994</ano>
</doutorado><livre-docente></livre-docente>
</professor>
<professor><nome>Neucimar Jerônimo Leite</nome><graduacao>
<curso>Eng. Elétr.</curso>
<universidade>UFPB</universidade><ano>1985</ano></graduacao>
<mestrado><universidade>UFPB</universidade><ano>1988</ano></mestrado>
<doutorado><universidade>Univ. Paris VI</universidade><ano>1993</ano></doutorado>

```
</professor>
<professor><nome>Paulo Cesar Centoducatte</nome><graduacao>
<curso>Eng. Elétr.</curso>
<universidade>UFES</universidade><ano>1982</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1992</ano></mestrado>
<doutorado><universidade>Unicamp</universidade><ano>2000</ano></doutorado>
</professor>
<professor><nome>Paulo Lício de Geus</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>ITA</universidade><ano>1976</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1979</ano></mestrado>
<doutorado><universidade>Univ. Manchester</universidade><ano>1985</ano></doutorad
</professor>
<professor><nome>Pedro Jussieu de Rezende</nome><graduacao>
<curso>Bach. Mat.</curso>
<universidade>Univ. Brasília</universidade><ano>1977</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1979</ano></mestrado>
<doutorado><universidade>Northwestern Univ.</universidade><ano>1988</ano>
</doutorado>
<livre-docente><universidade>Unicamp</universidade><ano>1996</ano></livre-docente>
</professor>
<professor><nome>Ricardo Dahab</nome><graduacao><curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1978</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1984</ano></mestrado>
<doutorado><universidade>Waterloo</universidade><ano>1993</ano></doutorado>
</professor>
<professor><nome>Ricardo de Oliveira Anido</nome><graduacao><curso>Eng.</curso>
<universidade>ITA</universidade><ano>1978</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1984</ano></mestrado>
<doutorado><universidade>Imperial College</universidade><ano>1989</ano></doutorado>
</professor>
<professor><nome>Ricardo Pannain</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>FEEC</universidade><ano>1982</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1988</ano></mestrado>
<doutorado><universidade>Unicamp</universidade><ano>1999</ano></doutorado>
</professor>
<professor><nome>Rogério Drummond Burnier Pessoa de Mello Filho</nome><graduacao>
<curso>Bach. Ciên. Comp.</curso>
<universidade>Unicamp</universidade><ano>1978</ano></graduacao>
<mestrado><universidade>Unicamp</universidade><ano>1980</ano></mestrado>
<doutorado><universidade>Cornell Univ.</universidade><ano>1985</ano></doutorado>
```

</professor>

<professor><nome>Tomasz Kowaltowski</nome><graduacao><curso>Eng. Elétr.</curso>
<universidade>USP</universidade><ano>1966</ano></graduacao>
<mestrado><universidade>Univ. California, Berkeley</universidade><ano>1970</ano>
</mestrado>
<doutorado><universidade>Univ. California, Berkeley</universidade><ano>1980</ano>
</doutorado><livre-docente><universidade>Unicamp</universidade></livre-docente>
</professor>

</corpo-docente>

<descricao-curso>

<credenciamento><paragrafo-tabulacao>
O Programa de Pós-Graduação em Computação, no Processo de Avaliação da CAPES, referente ao Biênio 1997/1998, recebeu conceito "5" (Equivalente ao conceito "A" nas avaliações de anos anteriores).
</paragrafo-tabulacao></credenciamento>

<integralizacao><paragrafo-tabulacao>
O curso de Mestrado em Computação será integralizado em um mínimo de 12 meses e em um máximo de 36 meses.
</paragrafo-tabulacao><paragrafo-tabulacao>
O curso de Doutorado em Computação será integralizado em um mínimo de 24 meses e em um máximo de 60 meses.
</paragrafo-tabulacao><paragrafo-tabulacao>
As disciplinas estão agrupadas em introdutórias, básicas, especializadas, tópicos especiais e seminários.
</paragrafo-tabulacao></integralizacao>

<disciplinas-programa>
<introdutorias>

<disciplina-pos>
<codigo>MO203</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Conceitos Básicos em Ciência da Computação</nome>
<observacao>Esta disciplina não tem seus créditos computados para efeito de conclusão do programa.</observacao>
</disciplina-pos>

</introdutorias>

<basicas>

<disciplina-pos>
<codigo>MO401</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Arquitetura de Computadores I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO403</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Implementação de Linguagens I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO405</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Teoria dos Grafos I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO406</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Linguagens Formais e Autômatos</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO409</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Engenharia de Software I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO410</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Bancos de Dados</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO416</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Introdução à Inteligência Artificial</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO417</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Complexidade de Algoritmos I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO441</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Computação Distribuída</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO442</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Processamento e Análise de Imagens</nome>
</disciplina-pos>

</basicas>

<especializadas>

<disciplina-pos>

<codigo>MO601</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Arquitetura de Computadores II</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO603</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Computação Gráfica</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO611</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Teleprocessamento e Redes</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO614</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Computabilidade e Funções Recursivas</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO615</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Implementação de Linguagens II</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO617</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Sistemas Operacionais Distribuídos</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO618</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Teste de Circuitos Digitais</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO619</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Geometria Computacional</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO620</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Engenharia de Software II</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO622</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Fatores Humanos em Sistemas de Computação</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO625</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Processamento de Linguagem Natural</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO626</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Representação de Conhecimento e Raciocínio</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO633</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Bancos de Dados II</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO637</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Complexidade de Algoritmos II</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO638</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Administração de Redes de Computadores</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO639</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Segurança de Redes de Computadores</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO640</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Biologia Computacional</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO641</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Projeto e Implementação de Sistemas Distribuídos</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO642</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Inteligência Artificial e Educação</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO645</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Projeto de Interfaces de Usuário</nome>

```
</disciplina-pos>
<disciplina-pos>
<codigo>MO646</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Construção de Interfaces de Usuário</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO647</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Introdução ao Projeto de Sistemas VLSI</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO648</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Projeto de Redes Multimídia</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO650</codigo><carga-horaria>0</carga-horaria>
<creditos>72</creditos><nome>Tese de Mestrado</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO669</codigo><carga-horaria>90</carga-horaria>
<creditos>06</creditos><nome>Estudo Dirigido II</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO800</codigo><carga-horaria>0</carga-horaria>
<creditos>144</creditos><nome>Tese de Doutorado</nome>
</disciplina-pos>
</especializadas>
<topicos-especiais>
<disciplina-pos>
<codigo>MO801</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Arquitetura e Hardware</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO802</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Linguagens de Programação</nome>
</disciplina-pos>
<disciplina-pos>
<codigo>MO804</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Teoria dos Grafos</nome>
</disciplina-pos>
```

<disciplina-pos>
<codigo>MO805</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Recuperação de Informação</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO806</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Sistemas Operacionais</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO809</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Computação Distribuída</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO810</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Inteligência Artificial</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO812</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Bancos de Dados</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO814</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Computação Gráfica</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO815</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Processamento de Imagens</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO817</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Semântica e Verificação de Programas</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO818</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Redes de Computadores I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO821</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Redes de Computadores II</nome>

</disciplina-pos>

<disciplina-pos>
<codigo>MO823</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Complexidade de Algoritmos</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO824</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Otimização Combinatória</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO827</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Engenharia de Software I</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO828</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Engenharia de Software II</nome>
</disciplina-pos>

<disciplina-pos>
<codigo>MO829</codigo><carga-horaria>180</carga-horaria>
<creditos>12</creditos><nome>Tópicos em Teoria de Computação</nome>
</disciplina-pos>

</topicos-especiais>

<seminarios>

<disciplina-pos>
<codigo>MO901</codigo><carga-horaria>45</carga-horaria>
<creditos>03</creditos><nome>Seminário de Computação</nome>
</disciplina-pos>

</seminarios>
</disciplinas-programa>

</descricao-curso>

<linhas-pesquisa><paragrafo-tabulacao>

As Dissertações e Teses poderão ser desenvolvidas principalmente nas seguintes áreas: algoritmos paralelos, ambientes e paradigmas de programação, análise de algoritmos, arquitetura de computadores, bancos de dados, biologia computacional, computação gráfica, métodos numéricos, criptografia computacional, engenharia de software, geometria computacional, groupware, informática na educação, inteligência artificial, interfaces homem-computador, linguagens de programação e compiladores, linguagens formais e autômatos, otimização combinatória, processamento de imagens, processamento de linguagens naturais, projeto e testes

de sistemas digitais, redes de computadores, sistemas de informações geográficas, sistemas distribuídos, sistemas operacionais, sistemas tolerantes a falhas, teoria dos grafos.

</paragrafo-tabulacao></linhas-pesquisa>

<normas-especificas>

<admissao><paragrafo-tabulacao>

Os candidatos interessados deverão encaminhar à Secretaria de Pós-graduação do IC ficha de inscrição devidamente preenchida, além dos documentos exigidos conforme o programa específico, entre os quais estão: históricos escolares; cartas de recomendação e curriculum vitae resumido.

</paragrafo-tabulacao><paragrafo-tabulacao>

As inscrições são aceitas até 31 de outubro para os candidatos que pretendem iniciar o programa em janeiro ou março seguinte. Os candidatos serão notificados sobre a aceitação ou não até o dia 15 de dezembro. Excepcionalmente poderão ser aceitos candidatos para iniciarem o programa em agosto. Neste caso, o prazo de inscrição encerra-se em 31 de maio e os candidatos serão notificados sobre a aceitação ou não até o dia 15 de julho. Poderá ocorrer o fato de a aceitação do candidato ao Mestrado ficar condicionada ao bom rendimento obtido em disciplinas introdutórias oferecidas em um curso de verão nos meses de janeiro/fevereiro ou ser aprovado em exame de admissão. O título de Mestre obtido no Instituto ou fora dele não dá ao candidato o direito de matricular-se automaticamente no Doutorado. Em casos excepcionais, candidatos sem o título de Mestre poderão ser aceitos para o Doutorado. Poderão ser aceitos alunos para cursar disciplinas isoladas. Os alunos nestas condições serão "Estudantes Especiais da Unicamp" e os creditos assim obtidos poderão ser convalidados no caso daqueles que se tornarem alunos regulares na Pós-graduação do IC. Inscrições para este fim devem ser feitas antecipadamente na secretaria, conforme calendário próprio do IC.

</paragrafo-tabulacao></admissao>

<bolsa-estudo><paragrafo-tabulacao>

O IC dispõe de um numero limitado de bolsas de estudo da CAPES e do CNPq. Estas bolsas são fornecidas aos alunos de Mestrado ou Doutorado aceitos, segundo critérios acadêmicos. Este tipo de bolsa também poderá ser solicitada diretamente pelo aluno e seu orientador à FAPESP.

</paragrafo-tabulacao></bolsa-estudo>

<esquema-cursos><paragrafo-tabulacao>

A Comissão de Pós-graduação (CPG) indicará para cada aluno de Mestrado ou Doutorado um orientador inicial, que poderá continuar como Orientador de Dissertação

ou Tese ou poderá ser substituído para esse fim por outro Docente. Este orientador inicial fixará, para cada candidato e de acordo com este, seu programa de estudos.

</paragrafo-tabulacao></esquema-cursos>

<requisito-titulos>

<mestre><paragrafo-tabulacao>

Os candidatos ao Mestrado devem ter concluído ou estar concluindo um curso de graduação, preferencialmente em Computação, Engenharia ou Matemática.

</paragrafo-tabulacao><paragrafo-tabulacao>

Para obter o grau de Mestre em Computação pelo Instituto, o aluno deverá:

</paragrafo-tabulacao><paragrafo-tabulacao>

1 - Conseguir, até 12 meses após o ingresso, a aprovação em disciplinas de Pós-graduação do Instituto, totalizando pelo menos 72 créditos, assim distribuídos:

</paragrafo-tabulacao><paragrafo-tabulacao>

Uma disciplina da área de Teoria da Computação, obrigatoriamente no primeiro semestre após o ingresso no curso - 12 créditos;

</paragrafo-tabulacao><paragrafo-tabulacao>

Uma disciplina na área de Sistemas de Computação - 12 créditos;

</paragrafo-tabulacao><paragrafo-tabulacao>

Uma disciplina na área de Sistemas de Programação - 12 créditos;

</paragrafo-tabulacao><paragrafo-tabulacao>

Outras disciplinas, à escolha do aluno e com aprovação do orientador - 24 créditos;

</paragrafo-tabulacao><paragrafo-tabulacao>

Dois semestres da disciplina Seminário de Computação - 6 créditos;

</paragrafo-tabulacao><paragrafo-tabulacao>

Uma disciplina de Estudo Dirigido entre o primeiro e segundo semestre do aluno no curso - 6 créditos;

</paragrafo-tabulacao><paragrafo-tabulacao>

2 - Definir, ao fim do Estudo Dirigido (cerca de 6 meses após o início no curso), o tema da dissertação; escrever um plano de trabalho (10-20 páginas) para a mesma; apresentar esse plano oralmente durante o Exame de Qualificação de Mestrado, e obter a aprovação desta última;

</paragrafo-tabulacao><paragrafo-tabulacao>

3 - Demonstrar capacidade de compreensão de texto técnico em inglês, em exame ministrado pelo Instituto, dentro do primeiro ano do aluno no curso;

</paragrafo-tabulacao><paragrafo-tabulacao>

4 - Completar e entregar a dissertação de Mestrado preferencialmente até 24 meses após o início do programa, apresentá-la oralmente perante uma banca de pelo menos três docentes (incluindo o orientador e um membro externo ao Instituto) e obter a aprovação desta.

</paragrafo-tabulacao><paragrafo-tabulacao>

Durante o transcorrer do primeiro semestre do aluno no curso, o aluno deverá escolher um orientador definitivo: um docente do Instituto que esteja disposto a supervisionar seu Estudo Dirigido, suas pesquisas no restante do curso e a elaboração da Dissertação.

</paragrafo-tabulacao></mestre>

<doutor><paragrafo-tabulacao>

Os candidatos ao Doutorado devem ter concluído um curso de Pós-graduação, de bom nível, em Computação, Engenharia ou Matemática. Em casos excepcionais poderão ser admitidos candidatos graduados, preferencialmente, em Computação, Engenharia ou Matemática sem o título de Mestre.

</paragrafo-tabulacao><paragrafo-tabulacao>

Além disso, um candidato ao curso de Doutorado somente será aceito se houver um ou mais docentes do Instituto que estejam dispostos a orientá-lo na sua área de interesse. Uma vez que a CPG estabelece limites para o número de orientandos por docentes, recomenda-se fortemente que candidatos ao Doutorado entrem em contato com possíveis orientadores, para confirmar sua disponibilidade e interesse, antes de se inscreverem para o programa.

</paragrafo-tabulacao><paragrafo-tabulacao>

Para obter o grau de Doutor em Computação pelo Instituto, o aluno deverá:

</paragrafo-tabulacao><paragrafo-tabulacao>

1 - Obter, até 24 meses após o ingresso no curso, a aprovação em disciplinas de Pós-graduação aprovadas pela CPG, totalizando pelo menos 72 créditos.

</paragrafo-tabulacao><paragrafo-tabulacao>

2 - Ser aprovado, no prazo estabelecido no Regulamento do Curso, nos exames de qualificação geral do Instituto, em pelo menos três das seguintes áreas:

</paragrafo-tabulacao>

<paragrafo-tabulacao>- Teoria da Computação (obrigatória);</paragrafo-tabulacao>

<paragrafo-tabulacao>- Sistemas de Informação;</paragrafo-tabulacao>

<paragrafo-tabulacao>- Sistemas de Programação;</paragrafo-tabulacao>

<paragrafo-tabulacao>- Sistemas de Computação;</paragrafo-tabulacao>

<paragrafo-tabulacao>

3 - Ser aprovado, até no máximo 12 meses após o exame de qualificação geral, no exame de qualificação específico sobre o assunto da tese, administrado por uma banca indicada pela CPG;

</paragrafo-tabulacao><paragrafo-tabulacao>

4 - Demonstrar capacidade de compreensão de texto técnico em inglês em exame ministrado pelo Instituto dentro do primeiro ano do aluno no curso;

</paragrafo-tabulacao><paragrafo-tabulacao>

5 - Completar e entregar uma tese sobre resultados originais de pesquisa, defendê-la oralmente perante uma banca julgadora de pelo menos cinco docentes (incluindo o orientador e 2 membros externos ao Instituto), e obter a aprovação desta.

</paragrafo-tabulacao></doutor>

</requisito-titulos>

</normas-especificas>

<instalacoes-equipamentos><paragrafo-tabulacao>

O campus dispõe de uma rede de fibra otica, que interliga os equipamentos da Unicamp e as redes nacionais e internacionais, via Internet. A ligação do IC a essas redes se dá através da Rede ANSP (Academic Network of São Paulo), gerenciada pela FAPESP. O Instituto conta com uma grande variedade de equipamentos computacionais. São diversos tipos de estações de trabalho SUN (Enterprise, Ultra, SPARC 1000, 20, 10, 4), PC Pentium, Macintosches e Silicon Graphics. Recentemente, adquiriu, através de projeto de pesquisa financiado pela FAPESP, computadores de alto desempenho de arquiteturas variadas, que vão desde máquinas multiprocessadas de memória compartilhada até máquinas paralelas do tipo "Beowulf". Alem

destes equipamentos, o IC dispõe de um bom numero de impressoras laser e de scanners de mesa.

</paragrafo-tabulacao><paragrafo-tabulacao>

Algumas das pesquisas em andamento fazem uso dos computadores (IBM SP 2) do CENAPAD (Centro Nacional de Processamento de Alto Desempenho) que se encontra situado dentro do campus.

</paragrafo-tabulacao></instalacoes-equipamentos>

<identificacao-disciplinas>

<legenda>

<paragrafo-tabulacao>

As disciplinas oferecidas pela unidade encontram-se a seguir identificadas. As informações são, na ordem em que aparecem, as seguintes:

</paragrafo-tabulacao>

<bullet>Código da Disciplina</bullet>

<bullet>Nome da disciplina</bullet>

<bullet>T - Total de horas aulas teóricas.</bullet>

<bullet>E - Total de horas aulas de exercícios.</bullet>

<bullet>L - Total de horas de laboratório ou de seminários.</bullet>

<bullet>S - Total de horas de trabalho de campo.</bullet>

<bullet>I - Total de horas de estudo dirigido ou de estudo em casa.</bullet>

<bullet>C - Total de créditos. Cada crédito corresponde a 15 (quinze) horas de atividades.</bullet

<bullet>P - Período mais provável da oferta da disciplina, de acordo com a convenção:</bullet>

<paragrafo-tabulacao>1 - 1° período letivo</paragrafo-tabulacao>

<paragrafo-tabulacao>2 - 2° período letivo</paragrafo-tabulacao>

<paragrafo-tabulacao>3 - qualquer período letivo</paragrafo-tabulacao>

<bullet>Os pré-requisitos (PR): exigidos para a matrícula na disciplina. AA200 - Significa Autorização da respectiva CPG.</bullet>

<bullet>A ementa descreve sucintamente o assunto relacionado com a disciplina. Em algumas disciplinas, principalmente aquelas relacionadas com Tópicos Especiais, as ementas serão oferecidas pelas Unidades de Ensino correspondentes, na época da oferta dessas disciplinas.</bullet>

<bullet>O livro onde se encontra o material básico (texto) pode também constar da informação de cada disciplina. No caso do material se encontrar em várias fontes, a lista bibliográfica será oportunamente fornecida pelo Professor Responsável pela disciplina.</bullet>

</legenda>

<ementas>

<disciplina>

<codigo>MO203</codigo><nome>Conceitos Básicos em Ciência da Computação</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Fundamentos matemáticos em análise de algoritmos (somas, probabilidade, notação assintótica e indução matemática). Estruturas de dados elementares. Algoritmos de Busca, ordenação e estatísticas de ordem. Grafos (representação de grafos e algoritmos básicos: buscas e ordenação topológica). Fundamentos de projeto. Processador básico. Conjunto de instruções. Pipelining. Hierarquia da memória. Caches. </ementa>

<bibliografia>Introduction to Algorithms. Cormen, T.H., Leiserson, C.E., Rivest. R.L. McGraw Hill, 1990. Introduction to Algorithms, a Creative Approach. Manber, U. Addison-Wesley, 1989. Computer Organization and Design: The Hardware/Software Interface. David A. Patterson e John L. Hennessy, Morgan Kaufmann Publishers, 1994. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO401</codigo><nome>Arquitetura de Computadores I</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Uma introdução avançada a arquitetura e organização de computadores. Tecnologias e perspectiva histórica. Medidas de desempenho. Conjunto de instruções. Unidades de aritmética e lógica. Projeto básico de um processador. Pipeline. Hierarquia da memória: cache e memória virtual. Dispositivos de I/O. Processamento paralelo. </ementa>

<bibliografia>David A. Patterson and John L. Hennessy. Computer Organization and Design; The Hardware/Software Interface, 1997, 2nd Edition. Morgan Kauffman. John L. Hennessy and David A. Patterson, Morgan Kauffman. Computer Architecture: A Quantitative Approach, 1996, 2nd Edition. Morgan Kauffman. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO403</codigo><nome>Implementação de Linguagens I</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Descrição formal de linguagens de programação. Análise léxica. Análise sintática. Geração de código. Sistemas de execução: blocos, procedimentos, recursão. Recuperação de erros. Ferramentas para construção de analisadores léxicos, sintáticos e semânticos. Construção de um compilador para uma linguagem exemplo. </ementa>

<bibliografia>Kowaltowski, T., Implementação de Linguagens de Programação, Editora Guanabara Dois, 1983. Aho A. V., Sethi R. e Ullman, J. D., Compilers - Principles, Techniques, and Tools, Addison-Wesley, 1986. Schreiner, A. T. e Friedman Jr., H.G. Introduction to Compiler Construction With UNIX, Pretince-Hall, 1985, Andrew W. Appel, Modern Compiler Implementation in Java, Cambridge University Press, 1988. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO405</codigo><nome>Teoria dos Grafos I</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Grafos, subgrafos, grafos orientados, famílias de grafos. ãrvores, caminhos, ciclos. Conexidade. Grafos eulerianos. Grafos hamiltonianos. Emparelhamento em grafos bipartidos. Coloração de arestas. Coloração de vértices. Conjuntos independentes. Teoria de Ramsey. Grafos planares. </ementa>

<bibliografia>R. Diestel em Graph Theory. Springer-Verlag, 1997 A. Bondy, e U. S. R. Murty. em Graph Theory with Applications. North-Holland, 1976.</bibliografia>

</disciplina>

<disciplina>

<codigo>MO406</codigo><nome>Linguagens Formais e Autômatos</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Palavras e linguagens. Gramáticas regulares e autômatos finitos. Não determinismo e generalizaçSes. Minimização de estados. Expressões regulares. Teorema da interação para linguagens regulares. Gramáticas livre de contexto e autômatons da pilha. Determinismo e ambigüidade. Teorema da interação para linguagens livres de contexto. Gramáticas sensíveis ao contexto e autômatons linares. Gramáticas e máquinas de Turing. Generalizações e restrições. Determinismo e algoritmos. Recursão e enumeração. Decidibilidade. O problema de Post. OperaçSes com linguagens. Transdutores e operações fechadas. </ementa>

<bibliografia>Hopcroft, J. e Ullman, J., "Introduction to Automata Theory, Languages, and Computation", Addison Wesley, 1979. Sipser, M "Introduction to the Theory of Computation", PWS Pub. Co., 1997. Kelley, D. "Automata and Formal Languages", Prentice Hall, 1995. Floyd, R. e Beigel, R. "The Language of Machines: An Introduction to Computability and Formal Languages", W. H. Freeman Co., 1994. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO409</codigo><nome>Engenharia de Software I</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Processos e modelos de desenvolvimento de software. Padrões de codficacão. Teste e verificação de programas. Distribuição e manutenção de software. Metricas de complexidade de software.</ementa>

<bibliografia>Pressman, R.S., "Software Engineering: A Practitioner's Approach", 4ª ed., McGraw Hill, 1997. Sommerville, I., "Software Engineering", 5ª ed., Addison Wesley, 1996. Ghezzi, C. Jazayeri, M. e Mandrioli, D., "Fundamentals of Software Engeneering", Prentice-Hall, 1991. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO410</codigo><nome>Bancos de Dados</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Introdução a sistemas de banco de dados incluindo modelos de dados, técnicas e teoria de projeto de bancos de dados, processamento de consultas e atualizações, esquemas para organizar e indexar arquivos e processamento de transações.</ementa>

<bibliografia>Ullman, J. D. Principles of Database and Knowledge Base Systems, volumes I e II, Computer Science Press, 1988 e 1990. Elmasri R. e Navathe, S. Fundamentals of Database Systems. Benjamin Cummings, 1994. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO416</codigo><nome>Introdução à Inteligência Artificial</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Histórico. Representação de conhecimento. Busca de informação e teoria de jogos. Inteligência Artificial Distribuída. Conexionismo. Aplicações: resolução de problemas, aprendizagem, processamento de língua natural, visão, robótica, sistemas especialistas e agentes inteligentes. </ementa>

<bibliografia>Russel, S e Norvig, P.; Artificial Intelligence: a modern approach, Prentice Hall, 1995. Winston, P. H., (1992) Artificial Intelligence, 3ª ed., Addison-Wesley. Firebaugh, M. W., (1988) Artificial Intelligence: Knowledge-Based Approach, Boyd and Fraser. </bibliografia>

</disciplina>

<disciplina>

<codigo>MO417</codigo><nome>Complexidade de Algoritmos I</nome>

<siglas teorica="60" exercicio="0" laboratorio="0" trab-campo="0" estudo-dirigido="120" credito="12" periodo="3"/><pre-requisito>AA220</pre-requisito>

<ementa>Modelos de computação e ferramentas/notação para análise de algoritmos. Indução matemática e projeto de algoritmos. Algoritmos gulosos. Programação dinâmica. Divisão e conquista. Algoritmos para ordenação e seleção. Algoritmos para problemas básicos em grafos. Reduções e NP-completude. </ementa>

<bibliografia>Cormen, Leiserson e Rivest. Introduction to Algorithms, MIT Press, 1990. U. Manber. Introduction to Algorithms. Addison Wesley, 1989. Brassard and Bratley. Algorithms. Prentice-Hall, 1996. Garey and Johnson. Computers and Intractability. Freeman, 1982. </bibliografia>

</disciplina>

</ementas>

</identificacao-disciplinas>

</corpo>

</catalogo-pos>

A.2 Documento DTD para o Catálogo da Pós-Graduação

```
<!- Declaracoes basicas ->
<!ELEMENT nome (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT ano (#PCDATA)>
<!ELEMENT paragrafo (#PCDATA | paragrafo-tabulacao | paragrafo)*>
<!ELEMENT paragrafo-tabulacao (#PCDATA | paragrafo-tabulacao | paragrafo)*>
<!ELEMENT bullet (#PCDATA)>
<!ELEMENT negrito (#PCDATA)>
<!- ->
<!ELEMENT catalogo-pos (capa, ficha?, calendario?, imagem?, apresentacao, corpo)>
  <!ELEMENT capa (cabecalho, titulo, subtitulo, ano) >
    <!ELEMENT cabecalho (#PCDATA | paragrafo)*>
    <!ELEMENT subtitulo (#PCDATA)>
  <!ELEMENT ficha (#PCDATA)>
  <!ELEMENT calendario (ano, mes-ano+)>
  <!ELEMENT imagem (#PCDATA)>
  <!ELEMENT apresentacao (titulo, diretor, diretor-associado, secretaria)>
    <!ELEMENT diretor (nome)>
    <!ELEMENT diretor-associado (nome)>
    <!ELEMENT secretaria (nome)>
  <!ELEMENT corpo (departamento*, comissao-pos*, introducao, corpo-docente,
                   descricao-curso, linhas-pesquisa, normas-especificas,
                   instalacoes-equipamentos, identificacao-disciplinas)>
    <!ELEMENT departamento (nome-departamento, membro+)>
      <!ELEMENT nome-departamento (#PCDATA | paragrafo)*>
      <!ELEMENT membro (nome, r)>
      <!ATTLIST membro ms (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9) #REQUIRED>
      <!ELEMENT r (#PCDATA)>
    <!ELEMENT comissao-pos (membro-comissao+)>
      <!ELEMENT membro-comissao (nome)>
      <!ATTLIST membro-comissao classificacao (Coordenador | Membro |
                   Suplente | Representante) #REQUIRED>
    <!ELEMENT introducao (#PCDATA | paragrafo-tabulacao | paragrafo)*>
```

```
<!ELEMENT corpo-docente (professor+)>
  <!ELEMENT professor (nome, graduacao, mestrado*, doutorado?, livre-docente?)>
    <!ELEMENT graduacao (curso, universidade, local?, ano)>
      <!ELEMENT curso (#PCDATA)>
      <!ELEMENT universidade (#PCDATA)>
      <!ELEMENT local (#PCDATA)>
    <!ELEMENT mestrado (universidade, local?, ano)>
    <!ELEMENT doutorado (universidade, local?, ano)>
    <!ELEMENT livre-docente (universidade?, local?, ano?)>
<!ELEMENT descricao-curso (credenciamento, integralizacao, disciplinas-programa)>
  <!ELEMENT credenciamento (#PCDATA | paragrafo-tabulacao | paragrafo)*>
  <!ELEMENT integralizacao (#PCDATA | paragrafo-tabulacao | paragrafo)*>
  <!ELEMENT disciplinas-programa (introdutorias, basicas, especializadas,
                topicos-especiais, seminarios)>
    <!ELEMENT introdutorias (disciplina-pos+)>
      <!ELEMENT disciplina-pos (codigo, carga-horaria, creditos, nome,
                observacao?)>
        <!ELEMENT codigo (#PCDATA)>
        <!ELEMENT carga-horaria (#PCDATA)>
        <!ELEMENT creditos (#PCDATA)>
        <!ELEMENT observacao (#PCDATA)>
    <!ELEMENT basicas (disciplina-pos+)>
    <!ELEMENT especializadas (disciplina-pos+)>
    <!ELEMENT topicos-especiais (disciplina-pos+)>
    <!ELEMENT seminarios (disciplina-pos+)>
<!ELEMENT linhas-pesquisa (#PCDATA | paragrafo-tabulacao | paragrafo)*>
<!ELEMENT normas-especificas (admissao, bolsa-estudo, esquema-cursos,
                requisito-titulos)>
  <!ELEMENT admissao (#PCDATA | paragrafo-tabulacao | paragrafo)*>
  <!ELEMENT bolsa-estudo (#PCDATA | paragrafo-tabulacao | paragrafo)*>
  <!ELEMENT esquema-cursos (#PCDATA | paragrafo-tabulacao | paragrafo)*>
  <!ELEMENT requisito-titulos (mestre, doutor)>
    <!ELEMENT mestre (#PCDATA | paragrafo-tabulacao | paragrafo)*>
    <!ELEMENT doutor (#PCDATA | paragrafo-tabulacao | paragrafo)*>
<!ELEMENT instalacoes-equipamentos (#PCDATA | paragrafo-tabulacao |
```

```
                paragrafo)*>
<!ELEMENT identificacao-disciplinas (legenda, ementas)>
  <!ELEMENT legenda (#PCDATA | paragrafo-tabulacao | paragrafo | bullet)*>
  <!ELEMENT ementas (disciplina+)>
    <!ELEMENT disciplina (codigo, nome, siglas,
                pre-requisito?, ementa?,
                bibliografia?, observacao?)>
      <!ELEMENT siglas EMPTY>
      <!ATTLIST siglas teorica NUMBER #REQUIRED
                exercicio NUMBER #REQUIRED
                laboratorio NUMBER #REQUIRED
                trab-campo NUMBER #REQUIRED
                estudo-dirigido NUMBER #REQUIRED
                credito NUMBER #REQUIRED
                periodo (1 | 2 | 3) #REQUIRED>
      <!ELEMENT pre-requisito (#PCDATA)>
      <!ELEMENT ementa (#PCDATA)>
      <!ELEMENT bibliografia (#PCDATA)>
```